Chapter 5

Adaptive Self-tuning : A Fuzzy Logic Approach

5.1 Introduction

Fuzzy logic approach provides the most simple and efficient way of implementing a feedback control mechanism in complex real-world applications [66] [67] like quality assurance in food processing, pharmacy, speed control of locomotives, washing machines etc. Application of fuzzy based control [68] is the most appropriate where the inputs and outputs are generally expressed in linguistic terms. For instance, in a washing machine, the weight of clothes being washed is expressed as light, medium and heavy. The weight of the cloth is measured and is mapped through a membership function to indicate its degree of belongingness to the three categories of weights. Depending on the degree of membership, the speed of the washing machine is chosen again as slow, medium and fast. The output is decided by a set of fuzzy rules and these have to be framed by the designer of the system using his understanding of the system that needs to be fine-tuned. Fuzzy-based tuning enables in building Policy-based [69] tuning systems.
The DBA expresses the tuning needs in linguistic terms. For instance, he/she might indicate that, if BHR is low then Buffer-Pool-size should be high depending on the current value of the BHR. He/She might also use an adjective in addition to the linguistic terms. For example, if the BHR may be expressed as very low instead of just low and the Buffer pool size to be very high instead of just high. As discussed in the earlier chapters, DBMS being a complex software system with its own set of tuning parameters having diverse set of values, application fuzzy is the most appropriate choice for implementing feedback-loop based tuning-action.

5.1.1 Fuzzy Inference Systems

The fuzzy rule-based systems, fuzzy expert systems are known as Fuzzy Inference Systems (FIS). The major function of this unit is decision making based on fuzzy rules that are of “IF THEN” type. These constructs might make use of conjunction operators like OR, AND, NOT while framing the fuzzy rules. The input to FIS may be fuzzy or crisp but the output will always be fuzzy. However, when FIS is used in a control system, the output must be crisp. Hence, the output from the FIS must be defuzzified to get the values in the crisp form. The working methodology of FIS is as under:

1. Inputs are fuzzified using different membership functions that include, triangular, trapezoidal and gaussian membership functions.
2. The rules are framed and stored in a database and this forms the knowledgebase for the FIS.
3. Finally output is defuzzified using appropriate steps.

There are two important types of FIS models, namely Mamdani model and Sugeno model[103]. In the former the output from FIS will be a fuzzy set. Hence, defuzzification is a must, whereas in the latter, the output is determined by a polynomial in the input variables and is first order. The output from Sugeno model will always be crisp and hence no defuzzification step is required. Though Sugeno model is computationally efficient and adaptive in nature, Mamdani model is preferred as it has wide acceptance, well suited for human input/output and is more intuitive in nature [61]. Hence, in the adaptive fuzzy tuning technique, Mamdani model is employed.
5.1.2 Membership functions for fuzzication

Membership functions are used to transform crisp input values of the real world application to fuzzy sets irrespective of the elements of the inputs, which are discrete or continuous.

Several membership functions have been proposed by experts in the field that include, triangular, trapezoidal, Gaussian etc. The membership functions are characterized by three important features.

1. Core: Core of a membership function is a region of the input where the membership value is exactly equal to 1. It means the inputs which fall in this category have complete membership.
2. Support: Support of a membership function is a set of input-values where the membership value is greater than zero.
3. Boundary: This part of the support where the elements that have partial membership that are non-zero.

The following notation is used to indicate the membership function values for the members of the input universe. If input $X=[x_1,x_2,\ldots,x_n]$ then the membership value of the fuzzy set $A$ is defined as

$$A = \{ \mu(x_i), x_i \in X \} \quad (5.1)$$
The membership values can be assigned using either intuition, inference or inductive reasoning, rank ordering, neural network methods. For all the input and output variables in the proposed self-tuning system, gaussian membership function[61] was used as it has smooth edges and hence there would be no abrupt changes in the estimated tuning parameter values.

Figure 5.2 Fuzzification of DB_Cache tuning parameter

5.2 Fuzzy Controlled tuning

Fuzzy logic is the most suitable choice for many control applications for the fact that fuzzy control systems are robust and can be tweaked easily to improve the system performance dramatically and most importantly they are much simpler in design to implement. Moreover, there is no need to measure the rate of change of the input parameters and the number of inputs and outputs are not limited to small number. In the proposed setup the input sensor parameters Buffer-Hit-Ratio(BHR), User-load(N) and Database Size(DBS) are extracted from the DBMS and are fuzzified before being fed to the Fuzzy Inference System(FIS).

5.2.1 Fuzzy Controlled Tuning Architecture

Figure 5.3 shows the generic autonomic computing architecture modified to implement self-tuning architecture for the DBMS wherein the FIS forms an important component of the entire self-tuning system. The FIS takes the sensor inputs namely the BHR, Userload N and Database Size as input to implement, Analyze and Plan modules of the MAPE based autonomic computing system. The inputs have to be fuzzified using appropriate membership functions [62] before being fed to the FIS. Similarly the output estimated by the FIS using fuzzy rules
from the Knowledge-base must be defuzzified before being used by the Execute module to fine-tune the DBMS.

![Diagram of the General Self-tuning architecture using Fuzzy Inference System](image)

**Figure 5.3 The General Self-tuning architecture using Fuzzy Inference System**

The FIS uses the fuzzy rules that are in the form “IF Then” to estimate the three tuning parameters namely, DB_Cache, Shared_pool and Large_pool. These three estimated parameters must be defuzzified to generate the crisp values before being used by the tuning module to fine-tune the system.

The defuzzification uses a mean max method to generate crisp output values. Figure 5.4 shows the complete fuzzy based self-tuning setup. The extraction module uses SQL commands and scripts to determine the BHR, User-load N and Database size DBS and feeds to the fuzzifier module.

**5.2.2 Fuzzy sub-system implementation**

Figure 5.4 shows the architecture of fuzzy controlled self-tuning system. The objective of this system is to analyze the DBMS, by proactively monitoring the performance indicators like buffer-hit-ratio, number of active processes and the database size that are showing signs of rapid growth and initiate control measure using fuzzy control. The inputs are BHR, N and DBS that are crisp and hence, have to be fuzzified before being acted upon by the FIS.
The FIS uses the fuzzy rules stored in the knowledgebase to decide on the values of the output variables of the fuzzy system.

In the fuzzy based tuning system the output variables are the tuning parameters themselves. The output fuzzy tuning values have to be defuzzified before being applied to the database system. The defuzzification uses centroid method to convert the fuzzy output variables to their crisp counterparts. Figure 5.5 shows the evaluation of the tuning parameter as the inputs vary using fuzzy rules.
5.2.3 Fuzzy Control Rules

Fuzzy control rules are based on general logical reasoning and based on programming language constructs namely the IF THEN control construct. Linguistic terms are used to describe the extent of tuning required.

Table 5.1 Fuzzy Inference Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>IF BHR is Best AND the user load is Low and DBSize is High THEN set BCS to low.</td>
</tr>
<tr>
<td>Rule 2</td>
<td>IF BHR is Best AND the user load is Medium and DBSize is High THEN set BCS to Moderate.</td>
</tr>
<tr>
<td>Rule 3</td>
<td>IF BHR is Good AND the user load is High and DBSize is High THEN set BCS to High</td>
</tr>
<tr>
<td>Rule 4</td>
<td>IF BHR is Low and the user load is High and DBS is high THEN set BCS to Very high</td>
</tr>
<tr>
<td>Rule 5</td>
<td>IF BHR is High and User load is Less and DBS is Small THEN set BCS to Low</td>
</tr>
<tr>
<td>Rule 6</td>
<td>IF BHR is Medium and and Userload is Medium and DBS is Medium THEN set BCS to Medium</td>
</tr>
<tr>
<td>Rule 7</td>
<td>IF BHR is Low and Userload is Medium and DBS is Large THEN set DCS to High</td>
</tr>
<tr>
<td>Rule 8</td>
<td>IF BHR is low and User load is Very High and DBS is Large THEN set DCS to Very High</td>
</tr>
<tr>
<td>Rule 9</td>
<td>IF BHR is High and Userload is High and DBS is Large THEN set SHP to Low</td>
</tr>
<tr>
<td>Rule 10</td>
<td>IF BHR is High and Userload is High and DBS is Large THEN set LRP to Medium</td>
</tr>
<tr>
<td>Rule 11</td>
<td>IF BHR is Low and Userload is High and DBS is Medium THEN set SHP to Medium</td>
</tr>
<tr>
<td>Rule 12</td>
<td>IF BHR is Low and Userload is High and DBS is Medium THEN set SHP to High</td>
</tr>
<tr>
<td>Rule 13</td>
<td>IF BHR is low Userload is High and DBS is Small THEN set SHP to low</td>
</tr>
</tbody>
</table>

(BCS ➔ Buffer Cache Size, DBS ➔ Database size, BHR ➔ Buffer Hit Ratio)

These rules must be carefully framed and most of the time require modifications till desired results are obtained. For instance, to find the new buffer caches size, some of the fuzzy rules could be formed as shown in Table 5.1. The values in figures 5.1 and 5.2 were fixed based on the user-load range of 1-100 and tuning parameter range of 0-1400MB respectively. The values in figure 5.5 were determined by the fuzzy-rules, the membership function used and the
defuzzification method employed. In this setup, Gaussian membership function was employed and mean of max method of defuzzification has been used.

5.2.4 Result and Analysis

The adaptive self-tuning method based on Fuzzy-control has been validated using standard workloads namely, TPC-C and TPC-H for its effectiveness in fine tuning the DBMS. Figure 5.6 shows the results wherein, the Fuzzy-control response-time is compared with auto-tuning feature of Oracle 10g under TP-C workload (SF=2). Due to rigid rule-based approach, the Fuzzy inference system slightly underestimates the tuning parameters values for the user-range(2-10) and hence, in the initial portion of the graph, the response-time is slightly higher than that of the auto-tuning method. However, for user-load beyond 50, the response-time is steady. The performance of Fuzzy-controlled tuning shows an improvement of 20.8% in response-time as compared to Auto-tuning feature of Oracle 10g.

![Figure 5.6: Response-time v/s Userload using Fuzzy based self-tuning](chart)

The fuzzy controlled tuning technique is also validated under TPC-E workload, SF=1 and the above graph in figure 5.7 shows and improvement of 28.7% over the built-in auto-tuning feature of Oracle 10g.
5.2.5 Fuzzy Controlled and Neural Network Controlled Tuning: A Comparison

Neural-Network-based tuning method shows much better performance over the entire user-load range of 2-100 as compared to fuzzy approach. However, it is much harder to implement Neural-network-based tuning system, as the construction of the training data-set is both time consuming and tedious. Further, the training data-set has to be framed for every kind of workload and scaling factor. On the other hand, Fuzzy based tuning technique is much easier to implement as it is simpler to frame fuzzy rules without the need to know the tuning impact of each tuning parameter on the response-time. However, it is important to know the range of each tuning parameter over which tuning is effective. Though, this method is simpler, it does not provide an improved response-time for all workload types over the entire range of the user-load.

Summary

In this chapter, a fuzzy-rule based adaptive tuning method has been presented. The input variables namely user-load, buffer-hit-ratio and the database size are fuzzified before being used by the Fuzzy Inference System. The FIS then uses the knowledgebase that is in the form of fuzzy rules to estimate the output fuzzy variables. The defuzzification generates the crisp tuning variables that are used by the tuner module to effect changes to the dynamic tuning parameters of DBMS. The method has been validated under two workload types namely TPC-C and TPC-E, and the results show an improvement of 18.52 and 28.7% improvement in
performance respectively. However, with TPC-C the performance is not at par with the Neural network based auto-tuning of the DBMS. This may be due to rigid rule-based nature of the FIS in estimating the tuning parameter values.