A Novel Analytics-as-a-Service System for Fuzzy Big Data

Rong-Kuan Shen\(^1\), Gwo-Jen Chio\(u\)\(^2\), Yi-Nan Lin\(^3\), Victor R. L. Shen\(^{4,5,*}\), Chan Hao Hsu\(^4\)

\(^1\)Department of Japanese Language and Literature, Shih Hsin University, Taipei City, Taiwan
\(^2\)Department of Electrical Engineering, National Formosa University, Yunlin County, Taiwan
\(^3\)Department of Electronic Engineering, Ming Chi University of Technology, New Taipei City, Taiwan
\(^4\)Department of Computer Science and Information Engineering, National Taipei University, New Taipei City, Taiwan
\(^5\)Department of Information Management, Chaoyang University of Technology, Taichung City, Taiwan

Email address
rkshen@cc.shu.edu.tw (Rong-Kuan S.), chuogj@nfu.edu.tw (Gwo-Jen C.), jnlin@mail.mcut.edu.tw (Yi-Nan L.), rshen@mail.ntpu.edu.tw (Victor R. L. S.), victor.rshen@msa.hinet.net (Victor R. L. S.), achilles1225@hotmail.com (Chan H. H.)

\(*\)Corresponding author

Abstract
Analytics-as-a-Service (AaaS) has become an important discipline in recent years. Many researchers are concerning more about collecting, collating, storing, and analyzing big data. This study aims to apply the theory of high-level fuzzy Petri nets (HLFPN) to the big data analysis platform. The platform can be applied to describe the analytical contents through natural language and to verify analytical processes by module. Also, it can be employed to generate Map/Reduce programs automatically and to be used for parallel computation, thus shortening analysis time. Finally, the experiments are successfully conducted to demonstrate the feasibility of our proposed approach.

1. Introduction
Since an ideal big data analysis system must provide complete services to customers, it should contain the following items: basic storage devices (e.g. structured or unstructured), data acquisition methods (e.g. streaming data or static data), analytical methods (e.g. real time analysis or learning analysis), user privacy, and data security. Also, a user interface can allow customers to control the analysis system correctly and to perform the data visualization.

Customers may come from different fields of expertise. For example, the medical doctors can analyze their patients’ healthcare records; the biologists can analyze DNAs; and the business owners can analyze the product sales and customer’s behavior. All of them need different data structures and analysis methods. A well- customized big data platform must have the ability to handle different kinds of data structures with various analytical methods.

Currently, this study intends to simulate Matlab fuzzy control systems [1] and to develop a high-level fuzzy Petri net (HLFPN) and a program generator for performing data analyses similar to the function in [2]. However, there exist some drawbacks in [2].
First, the more datasets are analyzed, the longer time it takes to perform the data analysis. Second, writing a Hadoop Map/Reduce program is more difficult and complex. Therefore, the HLFPN-based system on the cloud platform has been developed to overcome the above-referenced drawbacks.

Using the HLFPN model as a module and analysis tool has been shown to possess the following four advantages: 1) It offers more flexible learning capability because it is able to model both IF–THEN and IF–THEN–ELSE rules. 2) It allows multiple heterogeneous outputs to be drawn if they exist. 3) It offers a more compact data structure for fuzzy production rules so as to save information storage. 4) It is able to learn faster due to its structural reduction [3, 4].

Nowadays, no other platform can provide the users with easy operations. Some users do not know how to write an appropriate program. Users must first describe the contents of the work done by the system engineers. Then, the system engineers will help the users complete an analysis program. However, the communication between engineers and customers may be wrong, resulting in incorrect results. The main purpose of this study is to provide the users with a friendly interface, which can illustrate operating processes precisely, generating program codes automatically, reducing analysis time, and achieving customization.

The rest of the paper is organized as follows. In Section 2, the Hadoop Map/Reduce program is discussed and the basic properties of HLFPN theory are reviewed. The construction tools and system architecture are illustrated in Section 3. The experimental results and performance evaluation of the HLFPN-based big data platform are presented in Section 4. Finally, the conclusion and future work are remarked in Section 5.

2. Literature Review

In this section, the basic concepts of big data, IBM AaaS, and Hadoop are described. Also, the definitions of HLFPN are discussed.

2.1. Big Data

With the development of information and communication technology, the volume of big data has grown larger and larger. The size of datasets has currently reached the Petabyte and Exabyte scale in 2011 [2]. People have some ideas to process the big data [5-11], and to define Volume, Variety, Velocity, and Value as 4Vs. Volume is the key factor compared to the number of traditional datasets. Variety means the diversity of datasets. Velocity means processing data in an effective time. Finally, value is the true value of those data sets.

There are more and more research works on big data [12]. The most important sector is all about analysis methods, network transmission, and information security, privacy, and storage. There are a variety of issues and researches needed to be explored. In the future, big data analysis will become a service-oriented system which integrates the researches in various fields. A complete service platform is developed to provide customers with creating unlimited values.

2.2. IBM AaaS

With the advances of the Internet technology, the information explosion has occurred in the Internet with accessory devices. If we can fully utilize the information, we will be able to create new business opportunities and to make the existing services more efficient. However, it needs many professional experts in different fields to increase the data’s business value. The enterprise will cost much money if they invite those experts. Therefore, it is better to select an ordinary enterprise to outsource the professional enterprise.

![Figure 1. Analytics-as-a-Service platform proposed by IBM.](image-url)
The AaaS platform was first proposed by IBM [2], and its conceptual diagram is shown in Figure 1. The goal of the project is to study and develop an AaaS platform. However, they have faced several challenges in the process of creating one to provide various services, which include SLA (Service Level Agreements) definitions, QoS (Quality of Service) monitoring techniques, pricing, analysis and management of unstructured data, and business models.

In Richard K. Lomotev’s and Ralph Deters’ papers, those database systems, SQL and NoSQL [14], have been significantly improved for the storage method of structured and unstructured datasets; and provided a user-friendly API (application programming interface). They also proposed that the customized adaptive platform should be needed in the future.

2.3. Hadoop

Hadoop supports data-intensive distributed applications and provides an open source software framework with the issued Apache 2.0 license [15-18]. Due to the success of Google’s distributed file system in processing big data, the Hadoop Map/Reduce programs have drawn a lot of attentions, and received the praise from the industry and academics.

Hadoop supports applications on large clusters and building commodity servers, rather than relies on expensive proprietary hardware responsible for storing and processing datasets. Hadoop has many features suitable for big data management and analysis, shown as follows:

1. Scalability: Hadoop allows users to expand the hardware scale. The system can automatically redistribute datasets and computation jobs in order to accommodate hardware changes.

2. Cost efficiency: Hadoop brings a large scale of parallel computation to commodity servers, so that the hardware storage cost can be dropped down significantly. It makes big data keep the growth of a continuous supply of large-scale parallel computation.

3. Flexibility: Hadoop can accept any type of datasets from any source. A number of different types of datasets from different sources can be integrated with the Hadoop for further analysis. Therefore, many challenges in big data can be handled and resolved.

4. Fault tolerance: Lost data and computing failures are common cases in big data analysis. Hadoop can be recoverable due to data node failure or loss caused by network congestion and computing failures.

2.4. High-Level Fuzzy Petri Net

Petri net theory was proposed by Dr. Carl Adam Petri in 1962. The Petri net is a graphical and mathematical modeling tool, which is concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic. It can be used to model and analyze various systems [19]. However, along with the advances in the information technology, the description of a Petri net becomes more and more complex. Therefore, scholars conduct their researches with evolutionary Petri net theories one after another, such as colored Petri net [20], timed Petri net [21], fuzzy Petri net [22], high-level fuzzy Petri net [23-25], and so on. This paper adopted the HLFPN to make a decision on the fuzzy reasoning. It provides the characters of Petri net theory and fuzzy logic theory, which can be used to express fuzzy rules and to conduct fuzzy reasoning. Those basic definitions and fuzzy reasoning approach are presented as follows:

2.4.1. Definitions

1. Definition 1: The HLFPN is defined as an eight-tuple: $$HLFPN = (P, T, F, C, V, \alpha, \beta, \delta),$$ where

$$P = \{p_1, ..., p_k\} \quad \text{A finite set of places.}$$

$$T = \{t_1, ..., t_l\} \quad \text{A finite set of transitions.}$$

$$P \cup T \neq \emptyset$$

$$F \subseteq (P \times T) \cup (T \times P) \quad \text{Called the flow relation and is also a finite set of arcs, each representing the fuzzy set (i.e. fuzzy term) for an antecedent or a consequent; where the positive arcs (i.e. THEN parts) are denoted by} \rightarrow.$$  

$$C = \{X, Y, Z\} \quad \text{A finite set of linguistic variables, e.g.} \ X, \ Y, \text{and} \ Z, \text{where} \ X = \{x_1, x_2, ..., x_n\}, \ Y = \{y_1, y_2, ..., y_m\}, \ Z = \{z_1, z_2, ..., z_q\}.$$  

$$V = \{v_1, ..., v_l\} \quad \text{A finite set of fuzzy truth values known as the fuzzy relational matrix between the antecedent and the consequent of a rule.}$$

$$\alpha: P \rightarrow C \quad \text{Associations function, mapping from places to linguistic variables.} \ \alpha(p_i) = \{c_i\}, i = 1, ..., I, \text{where} \ C = \{c_i\} \text{ is a set of linguistic variables in the knowledge base (KB) and is the number of linguistic variables in the KB.}$$

$$\beta: F \rightarrow [0, 1] \quad \text{Associations function, mapping from the flow relations to the fuzzy truth values between zero and one.}$$

$$\delta: T \rightarrow V \quad \text{An association function, mapping from transitions to fuzzy relational matrices.}$$
2. Definition 2: Input and Output Functions

\[ I(t)=\{p \in P | (p, t) \in F \} \quad \text{A set of input places of transition } t. \]

\[ I(p)=\{t \in T | (t, p) \in F \} \quad \text{A set of input transitions of place } p. \]

\[ O(t)=\{p \in P | (t, p) \in F \} \quad \text{A set of output places of transition } t. \]

\[ O(p)=\{t \in T | (p, t) \in F \} \quad \text{A set of output transitions of place } p. \]

3. Definition 3: Membership Function

The mapping function \( \text{Mem}(p) : P \rightarrow [0,1] \) assigns each place a real value, where \( \text{Mem}(p) = \text{DOM}(\alpha(p)) \), DOM represents the degree of membership in the associated proposition and data tokens are available in a set of places, \( P \).

4. Definition 4: Max-Min Compositional Rule

In the HLFPN, \( \forall \) transition \( t \), \( V(t) = \min \) (fuzzy sets in \( I(t) \)); \( \forall \) place \( p \), \( V(p) = \max \) (fuzzy sets in \( I(p) \)). The Max-Min composition operator is denoted by \( \circ \).

5. Definition 5: Input Place, Hidden Place, and Output Place

In the HLFPN, \( \forall \) place \( p_i \in P \), if \( \forall t_i \in T \), \( p_i \notin O(t_i) \), then \( p_i \) is called an input place (IP) of \( t_i \); if \( \forall t_i \in T \), \( p_i \notin I(t_i) \), then \( p_i \) is called an output place (OP) of \( t_i \); otherwise, \( p_i \) is called a hidden place.

6. Definition 6: Heterogeneous Outputs

In the HLFPN, if the outputs possess different attributes, then we call them heterogeneous outputs.

2.4.2. Fuzzy Reasoning Algorithm

In this sub-section, a fuzzy reasoning algorithm (FRA) [24, 26, 27] is adopted to determine whether there exists a fuzzy relational matrix between the antecedent and the consequent of a fuzzy production rule.

INPUT: \( \text{Mem}(p_i) \forall p_i \in IP \), where \( IP \) denotes a set of input places.

OUTPUT: \( \text{Mem}(p_i) \forall p_i \in OP \), where \( OP \) denotes a set of output places.

3. The Proposed System

Linux is used to implement Eclipse as the main architecture, integrating other open sources such as PetriNetSim [29], jFuzzyLogic [30], Eclipse, Hadoop, Hbase and HDFS (Hadoop Distributed File System). Hadoop, Hbase, HDFS and Linux are formed as the underlying architecture. The other open sources will be discussed in more details. The architecture of our platform is depicted in Figure 2.

**Figure 2. System architecture.**
3.1. PetriNetSim

PetriNetSim [29] is a small Petri net simulator implemented in Java. Because of its comprehensive functionality, PetriNetSim can be used to store, read, draw, simulate, and execute Petri nets. The PetriNetSim interface is depicted in Figure 3.

![Figure 3. PetriNetSim interface.](image-url)

The proposed system is equipped with the functions for big data processing and fuzzy rule declaration. The eXtensible Markup Language (.xml) storage procedure in PetriNetSim is adopted to enable instant drawing, which is described in the subsequent sub-sections. The modified interface is illustrated in Figure 4.

![Figure 4. Modified interface.](image-url)
3.2. jFuzzyLogic

jFuzzyLogic [30] provides an API designed for editing fuzzy rules and membership functions (MFs). It enables to describe the desired MFs and fuzzy rules through the .fcl files, converting fuzzy rules and MFs to a curve diagram for examination, execution, and comparison. Therefore, jFuzzyLogic is a considerably comprehensive fuzzy logic editor. However, it lacks an interface, requiring the text files to be edited by menu. In this study, the HLFPN system is integrated with the editor and the functions of the editor are expanded, incorporating a user interface, which is further detailed in the subsequent sub-sections. The original architecture, text contents, and membership functions of jFuzzyLogic are illustrated in Figures 5, 6, and 7, respectively.

![jFuzzyLogic architecture](image1)

**Figure 5.** jFuzzyLogic architecture.

```
FUZZIFY inVar2
TERM poor := trian 0 2.5 5;
TERM good := trian 2.5 6 7.5;
TERM excellent := trian 5 7.5 10;
END_FUZZIFY
```

**Figure 6.** The file context in .fcl.

![Membership functions](image2)

**Figure 7.** The membership functions (MFs) done by jFuzzyLogic.
3.3. Eclipse

In computer programming, Eclipse is an integrated development environment (IDE). This study uses Eclipse to integrate with PetriNetSim, jFuzzyLogic, and Hadoop because it contains a base workspace and an extensible plug-in system for customizing the environment. Also, it is very convenient to develop a program by using the customized perspectives. Several tools are combined as one user interface. The development environment is illustrated in Figure 8.

![Eclipse development environment.](image)

3.4. System Architecture and Modules

As shown in Figure 9, this system is primarily composed of four modules, namely, variable declaration, input fuzzy rule, fuzzy rule parser, and program generation. This system contains the main database and the internal database. The main database stores the text files of fuzzy rules and the PN model entered by users. The internal database stores the targeted big data (primarily 1-Tera datasets). Because the HLFPN system does not include the data-uploading procedure, the datasets are stored in the internal database for parallel analysis. Finally, the resulting visualization module is used to display the visual results.

![System architecture.](image)
3.5. Variable Declaration Module

The aforementioned PetriNetSim and jFuzzyLogic are integrated in this module. Java swing is used to develop a user interface for defining initial parameters and entering membership functions. The related algorithms and operating processes are shown in Figure 10.

As shown in Figure 10, when a declaration is initiated, the .xml files must be opened and stored first. This module then searches for the fuzzy control language (.fcl) files. If the .fcl files do not exist, the module establishes the .fcl files with the same names as the .xml files. If the .fcl files exist, the module applies a parser to the .fcl files. Users can then define their desired parameters and MFs by using the graphical user interface (GUI), and combine the parameters and the columns of the internal database. The MFs can be entered as triangular, trapezoidal, bell-shaped, Gaussian, and Gaussian-based distribution. After the input process is completed, the system stores the variables declared in the .fcl files by users. During the parameter definition processing, a mechanism is implemented to avoid entering incorrect MFs, entering the same parameters, or defining repeated columns. If an error is detected, then a window displays the input error and the repeated parameters. When users complete variable declaration, the declared variables and MFs are stored in the .fcl files. The interface and functions of this module are depicted in Figure 11.
3.6. Fuzzy Rule Module

The operating processes are illustrated in Figure 12.

First, the .xml files are confirmed to be open. Second, the GUI with fuzzy rules is displayed, and the if-then rules are entered through the GUI. When a rule is entered, the module compares the variables in the rule with those in the .fcl files. If an error is detected, a window displays that the parameters have not been declared. If the entered rules are accurate, then the system stores the fuzzy rules in the .fcl files and proceeds to the fuzzy rule parser module. Its interface and functions are shown in Figure 13.

3.7. Fuzzy Rule Parser Module

The operating processes are illustrated in Figure 14.

This module succeeds the fuzzy rule module and receives an error dialog if users enter inaccurate or undeclared variables in the previous module. If the entered datasets are accurate, the module proceeds to six successive functions, the details and operations of which are described as follows:

[Step 1] Parse fuzzy rule: The fuzzy rules entered by users are analyzed, yielding variable values and the related fuzzy rule to acquire the types of Petri net (PN) modules.

[Step 2] Create a new PN module: Continued from the previous step, through the file-editing approach, the preset numbers, names, and connection modes of the .xml variables are converted into parameters.

[Step 3] Save a PN module as .xml files: The existing edited or unedited PN modules are stored in the initially opened .xml files, as shown in Figure 15.

[Step 4] Parse the .xml file: The parser is applied to the .xml files, and the identifications (IDs) are recorded. The IDs are the codes in the .xml files, namely, places, transitions, and arcs. They are the keys to identifying the .xml files. No elements are permitted to have the same IDs. If two or more elements have the same IDs, then the previous element may be overlapped by the next one, and the reading errors can be shown.

[Step 5] Add PN module to the .xml file: The .xml files edited from the previous step, including the rule parameters and names of variables, are copied and pasted into the text files in Step 3. During this processing, the IDs of the elements are all added by 1.

[Step 6] Open the .xml file again: Opening the edited text files reveals the newly added PN modules on the screen. Users can edit the PN modules according to their preferences.
Moreover, the values of the places in this module are all variables in the rules and are identifiable. The final results are shown in Figure 16.

In this module, the existence of the .fcl file is verified first. If the .fcl file exists, this module checks the rule file. The Map/Reduce program must be analyzed through fuzzy rules. If the .fcl file does not exist, a notification window appears, and the system switches to assess the loop. The processing steps are detailed as follows:

[Step 1] Check whether a rule is the final one: If not, proceed to Step 2; otherwise, skip to Step 5.

[Step 2] Create a Map/Reduce program: Similar to creating a PN module, generate a Map/Reduce program corresponding to the rule. The parser of the .fcl file is used to modify the analysis variables, the file contents represented by the variables, locations and algorithms of the files (e.g. max–min composition algorithm). The rules are then placed in the execution file folder.

[Step 3] Execute: This step executes the modified Map/Reduce program individually. The matrix multiplication (MM) algorithm is performed for each program executed. The MM of the current parameter and that of another parameter are executed. Then the result is stored in a new file for potential use by the next rule.

[Step 4] Wait for the results: The file name of a parameter is generated by executing the Map/Reduce program. The program receives the command to search for the file continuously in the file folder. If the file is located in the folder, then the Map/Reduce program is executed, and returns to Step 1.

[Step 5] When the rule parser ensures that all rules are converted into programming codes and executed, the system proceeds to the next module.
3.9. Result Visualization Module

Java jFreeChart is applied to ensure a simple dynamic data output. However, it does not enable a comprehensive search in the database. Currently, it displays only the results of each fuzzy rule, as shown in Figure 18.

![jFreeChart results](image)

Figure 18. jFreeChart results.

4. Experiment and Performance Evaluation

This experiment is intended to confirm that 1) the big data analysis platform is operated correctly; (2) the fuzzy membership functions can be set up on the platform; (3) the HLFPN model can be used and modularized; (4) the HLFPN model can be converted into Map/Reduce parallel programs; and (5) the results can be shown on the platform and compared with the stand-alone version of HLFPN.

4.1. Experimental Results

This experiment has applied the analytical approach cited from [4]. However, the number of datasets in the current study is approximately 20 times that analyzed in [2]. Furthermore, this study is divided into a normative HLFPN experiment (i.e. a control group) and a parallelized HLFPN experiment (i.e. an experimental group). The differences in consumed time, system usage, and analysis process between two groups are investigated. For their advantages and disadvantages, a normative HLFPN and a parallelized HLFPN are also compared with each other.

4.1.1. Normative HLFPN Experiment (Control Group)

The aforementioned procedure and architecture are used in this experiment, and the targeted datasets are stored in the internal database. The results mainly focus on the recording time. The actions and approximate time taken are shown in Table 1.

<table>
<thead>
<tr>
<th>Action Taken</th>
<th>Approximate Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Program (catch RSI, WMS, PSY values)</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Write Program (make MFs and fuzzy rule)</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Write Program (fuzzy reasoning)</td>
<td>40 minutes</td>
</tr>
<tr>
<td>Write Program (defuzzify)</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Analyze (2.5 Terabytes (TBs))</td>
<td>5340 minutes</td>
</tr>
<tr>
<td>Total Time</td>
<td>5450 minutes</td>
</tr>
</tbody>
</table>

As shown in Table 1, the normative HLFPN program may take time much longer than one hour. If the program includes a big data analysis, it may be much longer than five hours. Next, the same procedure continues to be used in our analysis platform.

4.1.2. Parallelized HLFPN Experiment (Experimental Group)

The actions and approximate time taken are shown in Table 2.

<table>
<thead>
<tr>
<th>Action Taken</th>
<th>Approximate Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare Variables (catch RSI, WMS, PSY values)</td>
<td>3 minutes</td>
</tr>
<tr>
<td>Declare Variables (define MFs and fuzzy rule)</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Generate Program (automatically run fuzzy reasoning)</td>
<td>1 minute</td>
</tr>
<tr>
<td>Generate Program (automatically run defuzzification)</td>
<td>1 minute</td>
</tr>
<tr>
<td>Analyze (2.5 Terabytes (TBs))</td>
<td>1800 minutes</td>
</tr>
<tr>
<td>Total Time</td>
<td>1815 minutes</td>
</tr>
</tbody>
</table>

As shown in Table 2, the parallelized HLFPN experiment uses this system to decrease the time in writing programs. Users can use the HLFPN-based system to declare the
variables, to define the membership functions, to enter the fuzzy rules and to generate the Map/Reduce programs automatically. The results from the parallelized HLFPN experiment compare advantageously with those from the normative HLFPN experiment.

4.2. Big Data with Domain Specific Language

In the paper [31], they built a big data platform with DSL (Domain Specific Language). The idea of the DSL platform is similar to ours. The differences between these two experiments are shown in Table 3.

| Table 3. The differences between DSL and HLFPN in big data analysis platform. |
| Functions                  | Big Data with DSL                                                                 | Big Data with HLFPN |
| Choose Data                | Type instructions below:                                                        | Choose the file folder |
|                           | “NETWORK_FILE”                                                                 |                  |
|                           | “SUBSCRIBER_FILE”                                                              |                  |
|                           | “BILLING_FILE”                                                                 |                  |
|                           | “CUSTOMER_COMPLAIN_FILE”                                                       |                  |
|                           | “COMPETITOR_FILE”                                                              |                  |
|                           | “PAYMENT_FILE”                                                                 |                  |
|                           | “CROSS_SALES_FILE”                                                             |                  |
| Declare Variables          | Location {loc1, loc2, loc3,…}                                                   | GUI interface    |
| How to Analyze?            | Type instructions below:                                                        | Use fuzzy rules   |
|                           | “CROSS_SALE”                                                                   | “if…then… else…if”|
|                           | “MOST_PROFITABLE_CUSTOMER”                                                      | to describe.     |
|                           | “MOST_PROFITABLE_TRAFFIC”                                                       |                  |
| Failure Check              | Type instruction “YES” or “NO”                                                   | Check the PN Model|
| Result Viewer              | Type instructions below:                                                        | Jfreechart        |
|                           | “output_FD”                                                                    |                  |
|                           | “output_FD.location”                                                            |                  |

In Table 3, the big data analysis platform with DSL provides a high level language for the users. However, it is not good enough. The big data analysis platform with HLFPN uses fuzzy membership functions and fuzzy rules to achieve the goal more easily. It doesn’t need to trap difficult instructions, and has more flexibilities to decide the way how to analyze.

4.3. Functional Comparison

In Table 4, the HLFPN-based platform and the IBM API are shown to have their own unique features.

| Table 4. Comparison between HLFPN and IBM API. |
| Natural Language Approach | HLFPN Analyzing Hadoop Program | IBM API Analyzing Hadoop Program |
| Programming Skill Needed  | Yes                           | No                               |
| Modularization            | Slightly Needed               | Strongly Needed                  |
| Automatic Conversion of Hadoop Program | Fully Automatic           | Semi-automatic                   |
| Source Code Cost          | Free                          | Charge                           |
| Acquisition of User’s Requirements | Easy to Self-define       | Hard to Self-define              |
| Efficiency                | Good                          | Fair                             |

75 users were invited to use this analysis platform. After explaining the operation procedure, each user answers a questionnaire. The final results for the satisfaction level obtained from the questionnaires are presented below.

1. This system has improved my understanding of fuzzy concepts. (95.15%)  
2. This system has improved my understanding of high level fuzzy Petri nets. (91.2%)  
3. I can use this system to define variables and membership functions. (96.3%)  
4. I can use this system to input fuzzy rules. (94.2%)  
5. I can use this system to check the fuzzy rules. (87.75%)  
6. I can use this system to generate a Map/Reduce parallel program automatically and to obtain the results. (90.15%)  
7. I can use this system to efficiently write an HLFPN-based program by myself. (91.5%)  
8. I can use this system to analyze big data, which is more time-saving. (91.5%)  
9. The GUI looks more user-friendly. (80.15%)  
10. I think this kind of analysis can help me analyze big data. (85.15%)  

According to the above results, if the satisfaction level 90% is set to be a successful criterion, then those items except (5), (9), (10) have achieved the desired goals. We need to improve this system to make the GUI become more user-friendly, add other functions to check the fuzzy rules, and change a way how to analyze big data. In summary, the high scores of satisfaction level have shown that our proposed platform is really promising and acceptable.
5. Conclusion

This study has applied the theory of high-level fuzzy Petri nets (HLFPN) for the big data analysis platform, which presents the following contributions: 1) It can be adopted to describe the analytical contents through natural language. 2) It can be used to verify the analytical processes by module. 3) It can be applied to promote fuzzy logic theory and to solve the problems with nonlinear equations. 4) It can be employed to generate the Map/Reduce programs automatically. 5) It can be used for parallel computation, thus shortening the analysis time. 6) It enables to show the results on GUI so as to meet data-visualization needs.

In order to explore more effective analysis methods, the proposed system will be improved further for the structured and the unstructured datasets. In the future, the streaming data analysis will be further investigated to provide better service interfaces, more functions, and more valuable contents.

Subject Area
System Modeling and Analysis of Fuzzy Big Data

Acknowledgements

The authors are very grateful to the anonymous reviewers for their constructive comments which have improved the quality of this paper. Also, this work was supported by the Ministry of Science and Technology, Taiwan, under grant MOST 103-2221-E-305-015.

References


