Automation of System Monitoring Based on Fuzzy Logic or Rules; Comparison of Two Designed Approaches with Regard to Computational Infrastructures

Włodzimierz Funika[1,2], Filip Szura[1,2], and Jacek Kitowski[1,2]

[1] AGH University of Science and Technology, ACC Cyfronet AGH,
ul. Nawojki 11, 30-950 Kraków, Poland
[2] AGH University of Science and Technology, Faculty of Electrical Engineering,
Automatics, Computer Science and Electronics, Department of Computer Science,
al. Mickiewicza 30, 30-059 Kraków, Poland
{funika,szura,kito}@agh.edu.pl

Abstract. This paper is focused on monitoring automation of distributed systems. In the presented research, AI-based approaches to distributed monitoring related to large distributed systems such as grids, were explored. In both presented concepts knowledge is used to make decisions regarding management actions, using rules and fuzzy logic. The first concept is an agent-less rule-based solution, implemented in a high-level monitoring system called Saude-Net. It allows to define actions for monitored resources, using a kind of expert system. The second solution, which exploits agents and fuzzy logic, is realized in a system called SAMM Compliant Agent. Both presented systems are capable of reacting to observed failures and of modifying their knowledge to better fit possible problems with resources. We also present a short comparison of the two concepts, and an analysis of their usage.

Keywords: system monitoring, automation, rules, artificial intelligence, fuzzy logic, fuzzy sets, error reporting.

1 Introduction

Since the PL-Grid infrastructure [1] is meant to be a country-wide compute- and data-intensive platform, facilitating its administrator's operations, e.g. due to the size of infrastructure, is one of the key issues. An installation of this size can be endangered by malfunctioning of its resources, which needs detection of their failures and fast reactions (responses) to them.

To accomplish this task it is necessary to provide support for monitoring and reacting to observed errors. To do this the system administrator is usually assumed to check any raised alerts and respond on them. It has to be done continuously. This paper presents two approaches aimed to help administrators in their daily work. These solutions use knowledge described by rules or fuzzy sets.

M. Bubak, T. Szepieniec, and K. Wiatr (Eds.): PL-Grid 2011, LNCS 7136, pp. 142–156, 2012.

Many existing systems like Autopilot exploit this type of mechanism related to Artificial Intelligence (AI) which is commonly considered as a very broad research area that focuses on "Making computers think like people" and includes disciplines such as Neural Networks, Genetic Algorithms, Decision Trees, Frame Systems and Expert Systems. Fuzzy logic can be also regarded as an AI supporting solution and is used in Multi Criteria Decision Analysis (MCDA) [2]. In the context of resource monitoring/management activities, fuzzy logic may be used to reply to the following question: *"which action is better for the values of the parameters under measurement (if these parameters are influencing decision-making, e.g. related to resource management)?"*. Both concepts, rules and fuzzy logic, are used in existing systems as their knowledge engine, not only in computer science but also in areas such as medicine, process control or financial service. Fuzzy logic may be used when the designer does not have enough knowledge to model the whole area. In these types of systems it is possible to use these sets and possess only a limited amount of information to begin with.

In contrast to fuzzy logic, rules are often used when the designer wants to model relationships with "IF ... THEN ...". In these situations it is possible to describe this by rules which are understandable for humans and machines. Rules allow to define a sequence of conditions which have to be fulfilled. The rules need to be well defined and should cover the whole considered area. The usefulness of expert systems made them very popular in medicine, computer science, robotics. On the market many expert systems are available, such as Gideon (used for diagnostics of diseases), CaDet (cancer detection), Thorask (selection of injured people). Expert systems are also becoming a noticeable tool for computer science itself.

In this paper we present two concepts which were developed as intelligent software which would be able to help the administrator of a large computational infrastructure to cope with everyday duties at lower cost. These systems are targeted to provide automation of system administrator's functions, e.g. data storage usage, services accessibility. Both of them try to address the discussed issue using knowledge mechanisms. Based on these two approaches we focus on building a monitoring tool aimed at handling faulty operations of infrastructure resources.

The rest of the paper is organized as follows: Section 2 gives a brief overview of some of the existing monitoring tools. Section 3 presents our approach to the issue of monitoring automation and our solutions implemented in two different monitoring systems. Section 4 presents some details regarding tests which were performed on both our systems. The last section summarizes the discussed solutions and shows ideas regarding future work.

2 Related Work

At present there are a lot of monitoring systems for large distributed systems, both commercial such as Intellipool Network Monitoring, and free such as Zabbix or Nagios. These applications were designed for different purposes. As mentioned in [3] "the goal of any monitoring system is to provide a centralized view of the

monitored resources so that systems administrators can analyze conditions and prevent or fix problems quickly and efficiently. Generally, a separate system is set up to host the monitoring system and is placed in a centrally-located point". All monitoring systems may be divided into two classes. One of them uses agents in monitoring, another group uses only commands and protocols. The former, agent-based approach is used in many of existing systems [4,5] because it allows to monitor various parameters and does not affect the network load too much. The latter solution, the agent-less one is used when the agents cannot be installed due to system restrictions. There exist many facilities which allow to monitor large distributed systems such as grids.

The first system which is going to be presented is Nagios [6]. It is often used as a basis for other monitoring facilities, e.g. EGEE Grid Infrastructure Monitoring. Nagios provides many plugins which allows for customizing for particular purposes. It is able to monitor many distributed system parameters, it also can be easily extended. Nagios monitors the status of host systems and network services and notifies the user of problems. The monitoring daemon runs intermittent checks on hosts and services one specifies using external plugins, which return status information to Nagios. It provides only a basic set of sensors, but custom sensors can be developed by using any existing programming language. Nagios functionality may be extended by many plugins. It is also possible to define system commands (scripts or executables) which are run when a host or service state is changed. This ability is called an "event handler" and it gives Nagios an ability to e.g. restart services, log event information, enter a trouble ticket into a helpdesk system. This tool is able to react to failures but its reactions have to be earlier well defined by the system administrator.

The second interesting system is Zabbix. This tool is similar to Nagios. Like the previous one this system is designed to monitor various network services. This solution uses agents to monitor statistics of Unix and Windows hosts. In contrast to Nagios the Zabbix monitoring tool is much easier to configure and manipulate. All these functions may be done easily during network monitoring using a simple and easy-to-use web user interface. It doesn't require any modifications in its configuration files like does Nagios in its basic form.

Another tool is Ganglia, which – as described in [7] – is designed to monitor large infrastructures. It is a scalable distributed monitoring system designed for high performance computing systems such as large localized clusters and even widely distributed Grids. It relies on a multicast-based listen/announce protocol to monitor the state within clusters and uses point-to-point connections among representative cluster nodes to federate clusters into a Grid and aggregate their states. Ganglia has the advantages of low per-node overhead, high concurrency and robustness. This tool is a very good scalable monitoring facility, but like the previous one this system does not allow to automate the monitoring, nor uses it knowledge to adjust its work to the observed situations.

A different monitoring solution is Autopilot. It is one of the first systems which uses knowledge to choose appropriate actions. As described in [8] Autopilot is a novel infrastructure for dynamic performance tuning of heterogeneous

computational grids based on closed loop control. This software, in contrast to the above, can take actions which can optimize data centers and distributed applications.

Another system which allows for automation is Intellipool Network Monitor [10] which enables to define a set of rules which will be used when an error will be observed in a monitored network. This system is designed for large enterprise networks and can monitor distributed systems. It uses predefined actions to solve observed problems. It also provides functions which are used to alert the system administrator about reported failures. A next system which allows to define reactions on failures is Hyperic HQ [11]. It allows to automatically perform simple system administration tasks to prevent and resolve a broad range of issues without human intervention.

The above overview shows that there are very few systems such as Autopilot which are able to make decisions on actions in an adaptive way. Most of the existing systems are designed only to monitor networks and show visual information to the system administrator – it is up to the administrator how they are going to tackle emerging problems. The tools are mostly used to inform about system (resource) failures. They allow to define simple actions such as service restart as a reaction on the failures reported, but this is done in form of static definitions. There also exist other monitoring tools, e.g. QStorMan [9]. This tool enables data-bound monitoring related to the infrastructure workload and users' activities.

3 Concepts of Our Solution

This section presents two concepts of automation of system monitoring [12] used: rule-based and semantic-oriented agent-based approaches. These concepts are implemented by the Saude-Net system [14] and SAMM Compliant Agent (SAMM-CA) [15], respectively, both of them are meant to react to captured system failures. Both concepts allow to manage monitored resources to optimize their work and usage by re-arranging a system configuration which is in the scope of the system administrator's responsibility. In this section we present further details regarding the concepts and relevant systems' architectures.

3.1 Saude-Net System

The first solution to be presented is an automation system which uses rule-oriented approach. This system allows to define multiple actions for observed monitored resources that we call *services*. This system is on top of external low-level monitoring facilities. In its current version it uses the Zabbix monitoring system to provide Saude-Net with data, and to perform appropriate actions based or this data when necessary. The back-end of the Saude-Net tool Zabbix was chosen because it is much easier to configure or manage than other tools of similar functionality. Zabbix is also an agent-based solution. There is no need to write a complex XML description of a new resource when it is added to the monitored resources list. Zabbix only requires a single line in the agent configuration.

Saude-Net was developed to automate monitoring using rules and actions. This system is allowed to extend or change predefined actions on-line as well as redefine rules on-line. These changes do not bring the system to a restart or a suspend – the user is unaware that the monitoring system has to modify, validate and reload its knowledge. The Saude-Net tool monitores the whole network using Zabbix as a low-level layer. When a failure is observed in one of monitored resources (host or service), Saude-Net tries to perform some actions. To accomplish this task it uses its knowledge, which is described by rules. At the beginning it creates a list of all possible actions for an observed failure. This list is generated with help of rules. For each action there is an associated preference value called Preference Value (PV). This value has to be set by the system administrator when the action is added to Saude-Net. This value should be from the interval between 0 and 1. The default value of PV is 0,5 as a center of the mentioned interval. The PV is used to determine which action is best. Saude-Net sorts the list of possible actions to choose the best one from all available ones. If there is only one action on the list it is chosen regardless of the value of PV. In the worst scenario an action with a value of PV close to zero might be used. The Saude-Net picks the first action on the list – it is an action with the highest PV value. In the next step Saude-Net performs a selected action, and after that it changes the PV value for all the actions which were added to the list including the action which was executed. This modification consists of increasing each PV by the Performance Tuning Value (PTV) using the following formula:

$$PV = oldPV + PTV \tag{1}$$

After all modifications the list is cleared for a next failure to be handled. PTV means "preference tuning value" which is dependent on the count of actions which were considered when an error occurred. This value is calculated according to the following formula:

$$PTV = \frac{N_s}{D} * const * \frac{I_w}{D} \tag{2}$$

where:

- D is the number of instances of any action in the created ranking. It is the count of elements which occur in the ranking returned by the knowledge engine.
- N_s is the number of services which are currently modified. They are defined as services which have to be optimized or repaired.
- I_w is one of the following values: 1 implies a situation when the action wins and -1 when it does not.
- *const* is a constant value which equals 1/700. This value was determined for the monitored infrastructure empirically through a series of tests. For this value the modification of the PV value was enough to manage actions. This value may be changed in configuration for different networks. It is dependent on the monitored system and should be set by the system administrator. The value 1/700 is the default value for which PTV was large enough to fit actions to the observed failures.

The above functions are used to choose the best action. They allow our system to learn which action is best and which one is worst. This is dependent on the statistics of the usage and on the history of each action. The preference value can be also changed by the system administrator when Saude-Net is working. The PTV formula is dependent on the count of all actions because when the system is able to choose many actions with similar PV values the modification should be smaller to avoid big errors.

A system which implements the above idea is designed as a modular tool, divided into three separate components: Saude-Net GUI, Saude-Net Server and Saude-Net Local-Monitor. These components are shown in Fig. 1 as layers.

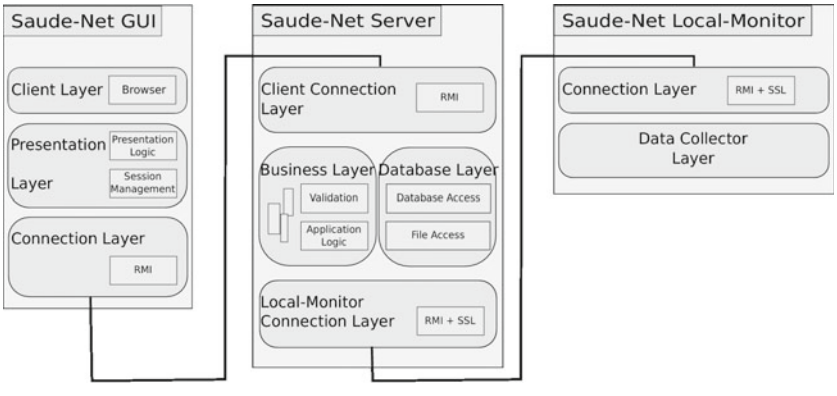

Fig. 1. Layers of modules of Saude-Net system

The first component is designed to allow the administrator to operate on rules and actions. It is responsible for simple validation and data presentation. The second module is the main component of the developed system, which is responsible for validation of rules and actions. It chooses the best action as a reaction to the failures reported. The last component is responsible for collecting of monitoring data. This data may be obtained from more than a single Zabbix monitoring server. This system is also able to exploit other monitoring tools. In order to do this it is necessary to adapt the implementation of the data collector to a low-level monitoring facility. Saude-Net is customizable by the system administrator.

Interactions between the modules as well as the basic architecture of the Saude-Net system are depicted in Fig. 2.

This system is developed as a tool on top of external low-level systems. It requires the system administrator to define both rules and actions for the whole distributed system under monitoring. Each type of monitored resource should have relevant actions associated with it. Nonetheless it is possible to deploy default actions which will be good enough for more than one type of resource, e.g. a function which is able to restart the appropriate resource, or a function which enables another instance of the monitored service.

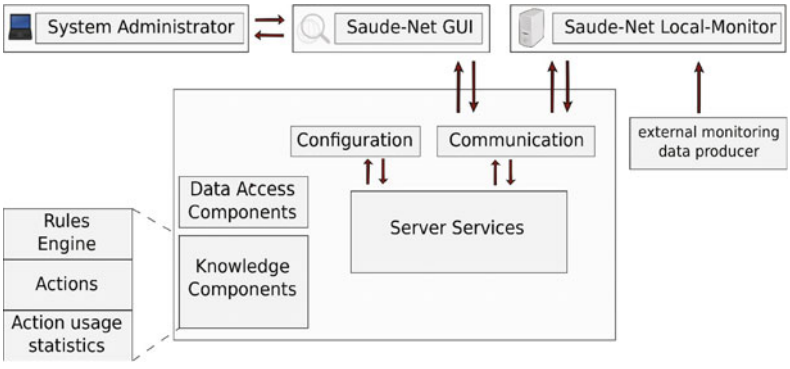

Fig. 2. Architecture of the Saude-Net system

3.2 SAMM Compliant Agent

The SAMM Compliant Agent (SAMM-CA) is the second system that is going to be presented. SAMM-CA is the solution for automation of system monitoring, based on semantics and agents. It is an extension to the Semantic-based Autonomic Monitoring and Management (SAMM) monitoring tool [16], which provides description of monitored resources in a flexible form of ontologies. SAMM is used as a low-level monitoring tool because it uses ontologies to describe monitored resources. It also allows to provide many statistics of system resources like CPU usage. SAMM exploits methods for reasoning using ontologies and data mining [13], which will be helpful for intelligent agents. SAMM-CA implements an agent-based approach for the automation of system management. The agent uses predefined actions and knowledge. Unlike Saude-Net, SAMM-CA uses knowledge described using fuzzy logic. This approach goes beyond the static nature of Saude-Net. The actions supported by SAMM-CA are associated with fuzzy sets. It allows to evaluate the purposefulness of actions described in terms of membership function with a few function types, like Gauss function or trapezoidal function. The use of fuzzy logic does not require to know the whole system model. It also allows to manage dynamically the borders of the membership function. Each set describes a single action. An action may be described by many sets. The measured parameters are used to define which action can be used for the observed failure. The agent allows to define the ranges of sets for which the associated action cannot be used. These sets are depicted in Fig. 3.

The borders of each fuzzy set are determined by the following set of formulas. For the left border there are:

$$\begin{aligned} y &= ax + b \\ a &= 1(x_1 - x_0) \\ b &= (-x_0)/(x_1 - x_0) \end{aligned} \tag{3}$$

The value x_0 describes the point where the set is starting with a value equal to 0 and the x_1 point defines the first point where the set possesses the

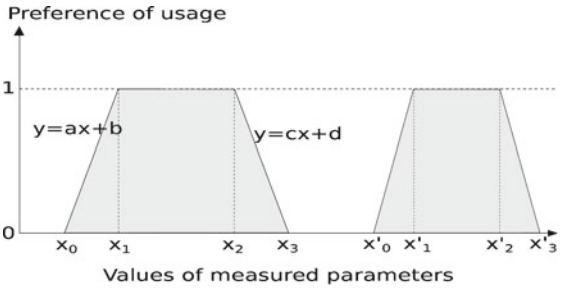

Fig. 3. Representation of fuzzy sets for actions in SAMM-CA

maximum value, which equals 1. For the right border the formulas are similar to the previous ones:

$$\begin{aligned} y &= cx + d \\ c &= -1(x_4 - x_3) \\ d &= x_3/(x_4 - x_3) \end{aligned} \tag{4}$$

The above formulas are used to determine the borders of these sets. The borders are modified while SAMM-CA works, whenever an action is performed. If this action ends with a positive result in the monitored system, the ranges of the set which describes that action may be extended. Otherwise this set may be reduced. Due to this fact the width of the range of the monitored parameter may be changed to fit the observed failures better.

The concept of SAMM-CA was to develop a solution which will enable resources management, especially decision-making, as close to the resource as possible. The developed agents possess their own knowledge. They are also allowed to exchange their knowledge and obtained information. The behavior of the agents is defined by associated modes. These modes are meant to enable the agents to send information about the failures observed, obtain information, and exchange their knowledge (see Table 1).

Table 1. Functions associated with agent modes

Mode	Functionality		
	Possess History	Learn	Exchange information
1	No	No	No
2	Yes	No	No
3	Yes	Yes	No
4	Yes	Yes	Yes
5	Yes	No	Yes

The global solution to the issue of the way SAMM-CA realizes its monitoring tasks is to optimize agents' knowledge by dividing it to separate entities. Each one of them learns separately but is connected with others, communicates with them and exchanges parts of knowledge. In some situations when one agent

is unable to resolve the observed problem it has to cooperate with others to respond. All agents can have an effect on the observed environment.

Fig. 4 depicting the main modules of SAMM-CA presents connections between modules and actions which are performed on these connections.

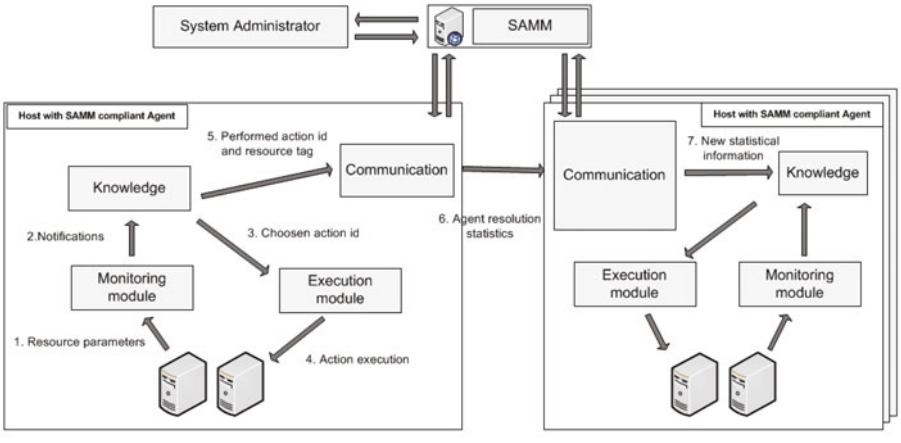

Fig. 4. Architecture of SAMM Compliant Agent

As mentioned before, each agent is able to perform its independent actions. One of the goals of this system is to provide a sufficient number of agents-experts to cover the whole scope of resources monitoring/management for a given infrastructure, each agent being an expert in a narrow area. The agents are assumed to exchange their knowledge and problems to be solved whenever necessary.

The agent-based solution presented in this paper is aimed at performing independent actions which are currently based on statistics and historical data. Each action is associated with a history of its usage. When a failure is observed in a monitored resource, SAMM-CA tries to use these actions whose fuzzy sets include the most recent value of the monitored parameter. The actions which are used most often may have larger widths of their fuzzy sets. These actions are more suitable for a wider scope of measured parameters and may be used when it is possible. When a value from within a wider range is observed, a relevant action can be performed and should be good according to the current knowledge of the agent.

The ranges of sets are closely related to the history of actions usage and they may be modified after each action execution.

Due to this fact these actions will be better suited for the observed problems – the sets are modified to better indicate, in which situations and for what value of the monitored parameter, the action may be performed.

3.3 Comparison of Presented Solutions

Table 2 presents a short comparison of both presented approaches, limited to the most important features.

Table 2. Comparison of the approaches implemented in Saude-Net and SAMM-CA

No.	*Saude-Net*	*SAMM-CA*
1	agent-less solution	agent-based solution
2	using rules engine	using fuzzy logic
3	centralized point of knowledge	knowledge distributed into separate agents
4	static expert system which requires administrator to describe all possible actions	dynamic knowledge, an agent is able to modify its knowledge and to add new fuzzy sets which will describe possible actions
5	actions have to be designed for the whole monitored system	actions may be defined individually for the separate areas of the monitored systems
6	developed for medium size systems	designed for all sizes of distributed systems
7	increase in the number of resources leads to modification in the central knowledge base	increase in the number of monitored resources implies modification of agents' knowledge or a need to start another agent for these resources

Both presented solutions were developed to automate monitoring of distributed systems such as grids. They were preliminarily tested on the PL-Grid test infrastructure. In the future it will be possible to run these solutions on grids for monitoring data storage. The SAMM-CA will be able to monitor a distributed data storage oriented infrastructure.

4 Evaluation Results

Both solutions presented above have their pros and cons. The first one, which is Saude-Net, may prefer only one action. In some cases it may be dangerous since there is a chance that a single action will be used all the time. Due to this fact, the monitored system might be unable to cope with some failures. It is caused by the lack of feedback from the monitored system.

The tests presented in this section were performed on a small infrastructure which contains only four hosts with LAN interfaces. On each host with Unix there was installed an Apache Server. This test infrastructure contains also one router and a switch. All four hosts were deployed in the same subnet. In all the tests this infrastructure was treated as one distributed system.

The behavior of the Saude-Net system through a sequence of failures is presented in Fig. 5.

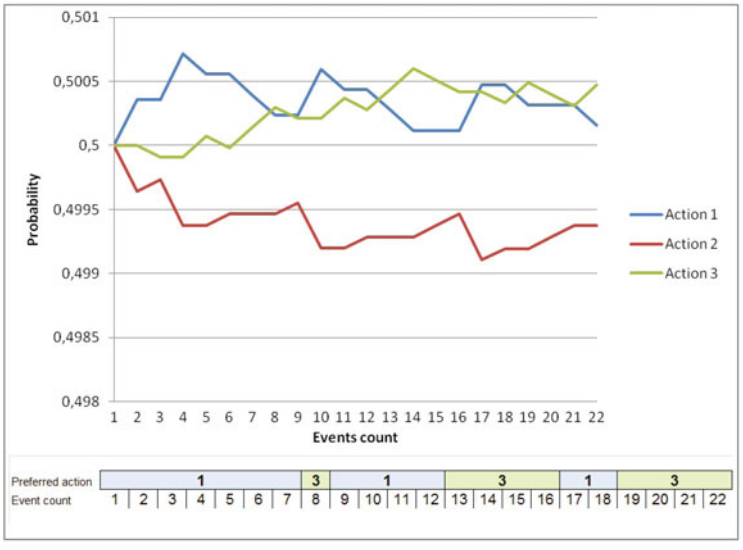

Fig. 5. Preference value course for three independent actions

The probability value in the chart shown in Fig. 5 defines which action is more suitable for the monitored resource. This chart represents available actions as a function of time for the monitored resource, which was the Apache Server, run on one of the experimental hosts. For this resource three possible actions were defined:

- Action 1 – server restarts on the same host.
- Action 2 – server starts on the alternative host.
- Action 3 – server is suspended for two minutes and after that this action has to restart the server on the same host.

All the actions were defined as Unix shell scripts. During the run of the system the preference values were changed due to the failures of two other servers for which the system administrator has defined only two of the above actions as available. During the test the preference values of all three actions were changed. On the presented chart the PV values of each action for Apache Server are shown. The system tried to manage its actions. It changed their preference values after it had tried to resolve each error notification. In the presented solution, action 2 is considered the worst one. Our system should use other actions but when there are no other choices it has to choose action 1. Saude-Net is able to cope with the actual situation because it not only uses the actions with the highest preference value but it can perform actions which are described as the worst ones if they are the only choice.

The second approach presented in this paper is SAMM-CA. Using this solution it is possible to perform one of the available actions and to fit them to the problem observed. In its current version it is able to exchange statistical

information about the observed failures and knowledge. When comparing these two solutions it might be noted that the first one, Saude-Net, uses rules which determine possible actions for each resource. This solution is not apparently as dynamic as SAMM-CA, which uses fuzzy sets. It allows to match a better action with a failure at a lower cost. Another difference is partially related to the system architecture: while Saude-Net uses the external monitoring tools to obtain data transmitted to a central point, which is a Saude-Net server, SAMM-CA is able to monitor resources by itself and to decide on actions locally. Saude-Net responses are slower than those performed by SAMM-CA, mostly due to communication costs. SAMM-CA uses local actions so they may be better suited for the monitored resources. These actions may be personalized for these resources like in Saude-Net, but in this case the system administrator is able to provide different actions for each agent. The second solution, SAMM-CA, was tested in the monitoring environment as well. These tests were dependent on the mode of each agent. In mode 1 the agents behave like a static expert system. When they are run in a different mode they are able to learn and communicate. During tests SAMM-CA was able to react on Apache Server failures like the Saude-Net system but its reactions were better suited when the borders of sets were fuzzy shaped. When the function which describes sets was more similar to a rectangle, the agent behaved like Saude-Net or any other rules-based expert system. The results collected from the agents are shown in Fig. 6.

This chart presents the test results where the x axis presents the count of the tested data to all data and the y axis shows the percentage of the correctly classified actions. This chart presents two tests. Each one was started with basic

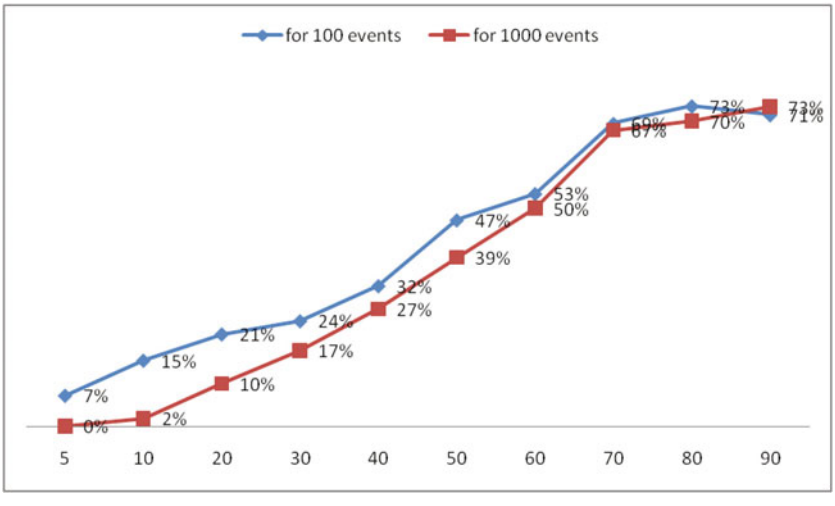

Fig. 6. Percentage of right classification of actions as a function of the size of the learning set with respect to the whole data set used for tests. The results for two independent tests are involved.

knowledge (the same for both of them). The results evidence that the developed system is able to learn and fit to the observed failures.

5 Conclusions and Future Work

The approaches presented in this paper were aimed at creating a solution for automation of system monitoring, i.e. decision-making what action should be taken to cope with a failure of a resource. The first solution is a rule based approach implemented in the Saude-Net system built on top-level of existing low-level monitoring systems, e.g. Zabbix. The second solution is a fuzzy logic-based approach called SAMM-CA. It uses agents to manage resources in a decentralized manner. It is implemented as an extension to the SAMM monitoring tool. In the Saude-Net system we introduced a rule-based knowledge engine. This facility is aimed to be used in the systems where the administrator can define multiple rules for each monitored resource under consideration. This tool is designed to learn which solution is the best for the observed situation, based on the concept of so called preference value. This system has a few drawbacks which may make it a little bit unstable due to the fact that the point of knowledge is central. The Saude-Net approach is an agent-less solution. There is a central point where data from external monitoring tools are first analyzed and afterwards this system can choose an action best suited to the observed failure.

The second approach is an agent-based solution implemented in SAMM Compliant Agent, SAMM-CA. Like the previous one, it extends another monitoring tool, specifically, the SAMM monitoring system which uses ontologies to represent knowledge on monitored resources. Unlike Saude-Net, which only uses rules, this system exploits the agent-based approach and fuzzy sets which are more flexible and should be more suitable for monitoring automation. This approach is not assumed to possess a centralized knowledge engine. Instead, it features distributed nature of its knowledge which allows to create a set of independent decision makers.

The SAMM-CA tool is able to better fit the monitored resources. It decides on actions locally so it is safer because it does not have to send an action description through the network. The Saude-Net system stores the whole knowledge in one central point. In large distributed systems this tool responds to captured failures rather slowly. In contrast to Saude-Net, the second presented solution is able to react much faster because its knowledge is distributed and is closer to resources. This allows to manage fragments of a system by one agent only.

To summarize the above, the SAMM Compliant Agent is a more suitable solution for the automation of monitoring of large grid infrastructures such PL-Grid: it uses fuzzy sets which underly a more flexible mechanism than rules, thus allowing for better action matching. On the other hand, the Saude-Net system is quite sufficient for the automation of infrastructure parts monitoring, where the communication costs can be outweighed by efficient event handling.

In the future the functionality of SAMM-CA is going to be extended. The system will be able to learn only one domain of the network resources. Another

modification – which will be implemented – is related to the agents: they will be able to exchange descriptions of the observed failures. This solution should make these agents more flexible. Each agent shall be a kind of expert system. Also in SAMM-CA an ability to manage its agents will be implemented. The system administrator will be able to choose which agent will be responsible for what kind of resources. For example it will be possible to configure one agent to monitor only storage devices and another agent to monitor only computational units or servers. The agents will be able to react only to a selected set of failures. If one agent will be unable to resolve an observed problem, it will delegate the problem to another agent.

References

1. PL-Grid project site, `http://www.plgrid.pl`
2. Figueria, J., Greco, S., Ehrgott, M.: Multiple Criteria Decision Analysis. Springer Science + Business Media (2005)
3. Stone, S.: Monitoring System Comparison. The Forbin Group (2007)
4. Cetnarowicz, K., Kozlak, J.: Multi-agent system for decentralized computer network management. In: Binder, Z. (ed.) Proc. MCPL 2000: Management and Control of Production and Logistics, 2nd IFAC/IFIP/IEEE Conference, vol. 1, pp. 469–474. Pergamon, Oxford (2001)
5. Cetnarowicz, K.: From Algorithm to Agent. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part II. LNCS, vol. 5545, pp. 825–834. Springer, Heidelberg (2009)
6. Comparision report on network monitoring systems (nagios and zabbix). Technical report, Malaysian Administrative Modernisation and Management Planning Unit (MAMPU) (2010)
7. Massie, M.L., Chun, B.N., Culler, D.E.: The ganglia distributed monitoring system: Design, implementation, and experience. Parallel Computing (2004)
8. Ribler, R.L., Simitci, H., Reed, D.A.: The Autopilot Performance-Directed Adaptive Control System. FGCS 18(1), 175–187 (2001)
9. Słota, R., Król, D., Skałkowski, K., Kryza, B., Nikołow, D., Orzechowski, M., Kitowski, J.: A Toolkit for Storage QoS Provisioning for Data-Intensive Applications. In: Bubak, M., Szepieniec, T., Wiatr, K. (eds.) PL-Grid 2011. LNCS, vol. 7136, pp. 157–170. Springer, Heidelberg (2012)
10. Intellipool Network Monitor, Intellipool AB (2011), `http://www.intellipool.se/4_0/doc/index.html` (access: June 15, 2011)
11. Hyperic HQ, System Monitoring Software, `http://support.hyperic.com/display/DOC/HQ+Documentation` (access: June 11, 2011)
12. Xiao Luo, X., Wang, Y.: The Research of Software Automation on Monitoring System. In: ISCID 2010 Proceedings (2010)

13. Han, J., Kamber, M.: Data Mining Concepts and Techniques. Academic Press (2001)
14. Funika, W., Szura, F.: Automation of decision making for monitoring systems. In: Bubak, M., Turała, M., Wiatr, K. (eds.) Proc. CGW 2010, October 11-13, pp. 164–171. ACC Cyfronet AGH, Kraków (2010)
15. Funika, W., Szura, F.: Agent-Based Monitoring Using Fuzzy Logic and Rules. In: Proc. 4th ACC Cyfronet AGH Users' Conference KU KDM 2011, March 9-11, pp. 28–30. ACC Cyfronet AGH, Kraków (2011)
16. Funika, W., Kupisz, M., Koperek, P.: Towards autonomic semantic-based management of distributed applications. In: Computer Science Annual of AGH-UST, vol. 11, pp. 51–63. AGH Press, Kraków (2010)