Włodzimierz Funika
Filip Szura

DATA STORAGE MANAGEMENT USING AI METHODS

Abstract *Data management and monitoring is an important issue in scientific computation. Scientists want to access their data as quickly as possible. Some experiments need to store a lot of data which have to be secure. By saying this we mean that this data can not disappear or be damaged also the data storage should be as cheap as possible. In this paper we present an approach to the automation of monitoring and management of data storage. We introduce a knowledge based system which is able to manage data, i.e., make decisions on migrating data, replicating or removing it. We discuss some of the existing solutions which are popular on the market. In this paper we aim to present our system which uses such AI techniques like fuzzy logic or a rule-based expert system to deal with data storage management. We exploit in this system a cost model to analyze the proposed solutions. The operations performed by our system are aimed to optimize the usage of the monitored infrastructure.*

Keywords data management, AI, fuzzy logic, rules, data storage

1. Introduction

Nowadays, the problem of data storage and data sharing is becoming more and more serious and important. In scientific computing it is often necessary to perform computations on large sets of data. Many of the performed experiments lead to the generation of huge amounts of data which have to be stored. Moreover, when defining the quality of computation it is often necessary to establish the level of access to data. Therefore, it is necessary to monitor the fulfilment of service parameters which are offered by storage devices.

In addition to scientific computing the growth of data and problems associated with their storage also occur in other areas like in industrial solutions. In both cases, besides the need to provide quick services in the context of data access it is important to fulfill other quality parameters which may be related e.g. to data security. These problems can be associated with fault tolerance, ensuring the transparency of continuous data access. All these aspects should be taken into account at the building, maintenance and implementation stages of storage infrastructures and tools which are aimed to provide data management.

In our research we exploit AI techniques e.g. fuzzy logic and rules due to the reasons that follow. Fuzzy logic control can be an excellent choice for systems with uncertain or highly complex dynamics. This is because fuzzy logic controllers require only a kind of working knowledge of system dynamics rather than a rigorous mathematical system model and are based on linguistic decision making expressions close to human thought patterns. Assuming a knowledge of the system, fuzzy controllers are relatively easy to construct, requiring only the creation of a set of linguistic "if-then" expressions that are the basis of the control and tuning of the rules. Our target environment is a dynamically changing infrastructure in which the parameters of each storage device are dynamically changing also the users requests (related to data upload or download operations) are unpredictable, due to this reason we need to use a control which is able to deal with this problem. Moreover, during tests we can observe that the engine which involves AI algorithms is able to perform actions which are appropriate for the monitored situation. These algorithms are able to adapt to situations which occur most often. Due to that the system can react more adequately to the situations observed in the future. The usage of AI techniques enables us to train the system to be prepared for the possible situations that may be observed and may be different for infrastructure elements which are assumed to be heterogeneous. It allows us to prepare the knowledge and the behavior of our system to situations which may occur in any test scenario or in future changes to the infrastructure. Artificial intelligence methods enable to look for the reason of the failures and to protect us from further ones. Summing up exploiting AI methods is a real help in problem solving and problem inspecting in a highly dynamical heterogeneous data storage infrastructure.

This paper is organized as follows: Section 2 "Related Work" gives an overview of some available solutions for infrastructure monitoring. This section presents some techniques of AI as well. Section 3 introduces our concept of AI-based data management. Section 4 presents the implementation of our idea and its functioning on a real example. Section 5 gives some results on the testing of our system. Section 6 concludes our work and presents some plans for the future research.

2. Related work

In our research we analyzed some existing algorithms and tools which relate to data management [3], [2]. Some of them use techniques of artificial intelligence to accomplish this task. Most of them are able to interact with storage infrastructure to perform appropriate actions to replicate, modify or delete a storage of data [11].

The authors of [6] present their work on management issues. The main goal of that work is to recognise the symptoms of the proactive management problems. A problem in this issue is recognized using artificial intelligence techniques. Artificial intelligence is used for network management. It is aimed to develop a proactive management. The authors created an expert system which exploits rules and knowledge structures.

Another algorithm is called FIRE [4]. This algorithm is designed for data grids which provide large-scale geographically distributed data resources for data intensive applications such as high energy physics and bioinformatics. This approach is able to perform replication at the moments when the task being executed on the grid requires data that are not available locally and there is some free space on the nearest storage device. Otherwise, the needed data is accessed remotely. In this approach the authors take into account the cost of replication as well as the cost of accessing the required file remotely. The FIRE approach seeks dependencies between the stored data and tasks on each node. When local data are not accessed then replicas may be removed. When replicating the data on the grid, the algorithm FIRE performs a comparison of possible data sources including the cost of replication from a given source as well as the cost associated with the length of the queue of tasks assigned the given source. This approach is aimed to optimize the usage of connections between sites during computation. It performs actions to accelerate user computations but it does not account for specific user requirements which relate to data storage.

The next monitoring solution which we want to present is Autopilot [14]. This is one of the first systems which uses knowledge to choose appropriate action. Authors describe Autopilot as a novel infrastructure for dynamic performance tuning of heterogeneous computational grids based on closed loop control. This software is able to take actions which can optimize data centers and distributed applications.

A next strategy to utilize an artificial intelligence is an anti-virus detection which is shown in the solution described by Xino-bin Wang et al. [16]. They present an approach to virus detection which is done by using data mining and neural network. The proposed solution is able to detect virus invasion in real time and enlarge the security management capacity of system administrators to enhance the integrity of the infrastructure of information security.

Another important approach which is available for data management is QStorMan [9]. This is a toolkit which uses an explicit definition of non-functional requirements regarding storage resources, usage of semantic descriptions of the available storage infrastructure and active monitoring data. This solution is developed for grids. It allows us to define QoS parameters for three main classes of entities which are: Virtual Organizations, users and applications. This toolkit allows the user to influence the distribution of their data by specifying QoS requirements. This solution uses a knowledge base subsystem (GOM) to find the best storage resource.

Another solution which is connected with QoS requirements is presented by Prodan et al. in [12]. This paper presents an approach to fulfill QoS in MMOG (massive multiplayer online game) games. It is used to predict the count of users on a separate site. It is also possible to predict the count of users in the whole game. Games which are used in this approach are divided into separate zones. For each one the count of future users are predicted. This prediction is done by the use of neural network where as an input layer is used a count of users in previous steps. The authors use for their system two kinds of neural network: one is a feed-forward network and the other is modified three layer perception.

In addition to the data oriented monitoring tools [17] we analyzed which of AI techniques like fuzzy logic, neural network or rules may be applicable to the data management issues. At first fuzzy logic [8] as lower-level knowledge was considered. Solutions like JFuzzyLogic offer many useful functions for knowledge recognition, but did not meet our needs fully. Other knowledge oriented libraries taken into account were JBoss Drools and Jess. Both of them have a similar functionality which was important for our needs and provided enough features for our purposes. Further comparing of those libraries enabled us to make a choice in favor of JBoss Drools [1], due to its better documentation and larger user community. Hence, we are using JBoss Drools as part of our knowledge engines.

3. Concept of the presented solution

Data storage management is a difficult issue. This is important that the stored data should be placed on resources which are able to fulfill users requests. Also, it is necessary to optimize the usage of resources and minimize the cost of their usage. Due to that it is important to develop a system which will allow us to deal with this task. In order to enable effective decision-making it is often necessary to get custom metrics which characterise the behavior of the objects under monitoring, like those described in [7, 17].

Due to the problems discovered during research which refer to data storage, we aim to develop a system that will allow us to manage the monitored infrastructure and will be able to store user data while fulfilling his/her quality requirements. In this context we consider the requirements related to the read or write speed or the number of replicas which have to be created to fulfill the user security requirement. In order to minimize administrative work the system under discussion is intended to exploit some AI methods to determine a relevant solution to the observed situation, like a performance flaw, in the infrastructure under monitoring.

We introduce two AI methods. The first one is based on fuzzy logic and is able to cope with uncertainty which may be connected with the observed situation. The second AI type exploits rules which define the behavior of the system under discussion and its actions on the situations observed. The presented approach takes into account user's QoS parameters with regard to storage resources and the cost model associated with storage data placement.

The cost model is often implemented in the monitoring software. This is done to optimize the usage of resources. The complexity of the cost model may be different in various systems. Some of them use only the cost of replica management while others take into account the cost of data transmission. The cost model exploited in our system is based on the cost of the maintenance of replicas, their count and also takes into account the cost of data transmission. The module which contains the cost model is responsible for collecting the actions proposed by the system. After that we use the cost model to pick the cheapest one. The established cost model is discussed in more details in Section 3.3.

In this section we intend to present our solution and its parts which are connected with AI and a cost model. We present AI techniques which we introduce to our system and a cost model. Both AI techniques are used to prepare a list of acceptable solutions when the system observed that something may be optimized. After that phase the lists of solutions (sequence of actions to be lunched) from both AI techniques are linked. Next, we use a cost model to identify which of the proposed solutions is the cheapest one and fulfill the requirements. A detailed description of the use of our system is presented with two use cases which were presented in Section 4. At the beginning of this section we are going to introduce our knowledge engine which is based on fuzzy logic. In Section 3.2 we present an expert system which we use and which operates on rules. The third part of of this section is dedicated to the cost model which is part of the module which is responsible for comparison of the proposed solutions.

3.1. Fuzzy logic

As the basis of one of the knowledge engines we used fuzzy logic. In this knowledge engine we present a single observed value like read speed as the independent dimension of our logic. We create a multidimensional area in which we create sets that are related to individual solutions. Each solution can possess a few of these multi-dimensional sets. At the beginning these sets can be modified, created, or even deleted. When our system detects a dangerous situation it starts to analyze its knowledge to choose a proper solution. When it finds an appropriate solution this solution can be performed. After that the system tries to observe the results of applying this solution. If it was appropriate for the values measured the system enlarges the set which describes this solution, otherwise the set is reduced.

3.2. Rule-based expert system

The second knowledge engine operates on rules. At the beginning we define a set of rules which are used as a reaction to an observed situation. These rules are created based on expert knowledge. During the system's operation we allow to automate modifying some of the rules. This modification is based on the modification of parameters' range that define which rule is correct. These parameters can be modified when the system detects that the performed action, which was described by the rule changed the behavior of the system.

3.3. Cost model

The last important part of our system is the module which provides a cost model. This part of our system is created to compare the proposed solutions. We combine the lists of solutions which were returned by the knowledge engines described above. After that each solution is checked. We calculate the cost of the described actions, e.g., replica creation. Also, we calculate the cost of a current situation without any actions including a cost of communication between sites. As the outcome of this module we obtain the best solution whose cost is minimal. The returned solution is performed on the infrastructure under monitoring.

3.4. Test environment

We tested our solution on a simulated environment. As the data input for our system we used a simulator for monitoring whose output is similar to the SMED [15] application which is part of QStorMan which will be used in the future as a source of monitoring data. During the current research we try to test and check the correctness of our knowledge engines. Due to that the solutions (actions) proposed by these engines were very simple and were aimed to present their behavior. The performed tests were divided into two main use-cases presented in Section 4. Both of them were performed using the same environment but in the use-case in Section 4.1 we check the system responses on user requests. Due to that in this scenario the system was dependent on the user. The test environment is depicted in Figure 1. It presents the key feature of this environment. They are: *Knowledge Engine* which is responsible of action selection, *Site manager* an independent application which is responsible for managing replicas which are stored locally. The third important part is a simulation of user actions on the system which are used to test our knowledge engines.

4. Use cases

In this section we are going to present our idea using two use-cases performed on the environment described in Section 3.

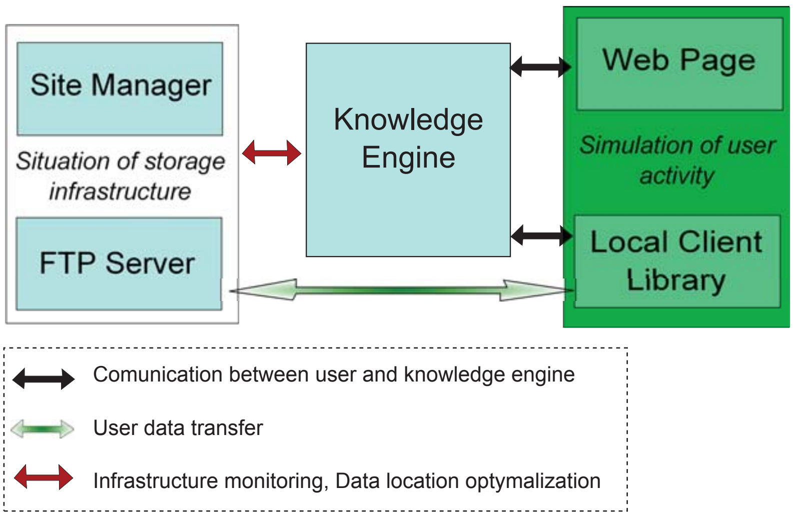

Site Manager
Situation of storage infrastructure
FTP Server
Knowledge Engine
Web Page
Simulation of user activity
Local Client Library
Comunication between user and knowledge engine
User data transfer
Infrastructure monitoring, Data location optymalization

Figure 1. Modules in the proposed system and relationships between them.

4.1. Response on a user request

This use case related to the situation when the system responds to user requests is depicted in Figure 2. At the beginning the user has to log into our system. When a client decides to upload or download a selected file the client interface passes relevant requirements to *KnowledgeEngine*. After that the request is analyzed by the running knowledge engine that may use fuzzy logic, rules or a neural network, which is already under testing. It is possible to analyze the user request by more than one knowledge engine. All these engines receive the latest monitoring data from an external monitoring system. All the received information is used during the knowledge analysis. Each of the knowledge engines tries to find the best or a group of possible best solutions (action(s)).

As an action we describe the function which is performed to optimize the storage of files, e.g., data replication or data migration.

After that it passes a list of these solutions to the *Solution Analyzer* which tries to pick the best action that will be as good as possible and will be cost effective. In this step the analyzer uses the information on the available files and their location. Also, this component uses a cost model to define which location will be the best one to use and what action should be performed.

The adopted cost model comprises two main parts. The first one is related to the cost of the storage of the file or its replica on the particular resource. This cost depends on the specification of the resource which holds a file or its replica. When our system tries to add a new replica to the monitored infrastructure this cost is calculated for each resource which is proposed in a solution coming from one of the knowledge engines. This model also takes into account the cost of an SLA contract break.

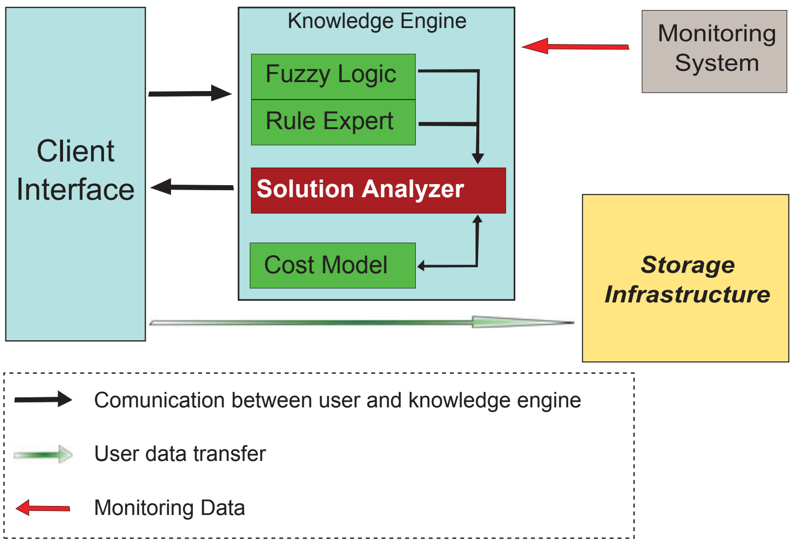


Figure 2. Communication between modules during responding to the user's request.

After that a response for the user is generated. This response is dependent on the user operation on a file. When he / she wants to add a new file the system returns a most suitable location for that file; when he/she wants to download an existing file the location of a replica which is the best candidate is returned for him/her. If the user wants to upload a new file to the infrastructure this information is stored in a database by the file replica management component. When the users realise their upload request, on the completion of the user's request the system is notified about this fact.

4.2. Infrastructure monitoring and detection of performance flaws

The second use case is related to the management of data storage infrastructure. This function is done without any user or administrator participation. The system collects monitoring data, after that it tries to predict the behavior of the monitored system and afterwards it searches for a possible failure of the monitored resources or for the probability of that the user QoS parameters will not be met. If one of the above events occurs in the monitored infrastructure the system under discussion tries to find a solution which can help and is cost effective. By saying this we mean that the cost of the manipulation of the data will be lower than the cost connected with SLA contract break. In Figure 3 we present the same monitoring infrastructure as in Figure 2, in this case there are no user requests and the system looks through not only the knowledge of our system but also the database that stores information about the replicas which reside on a particular device.

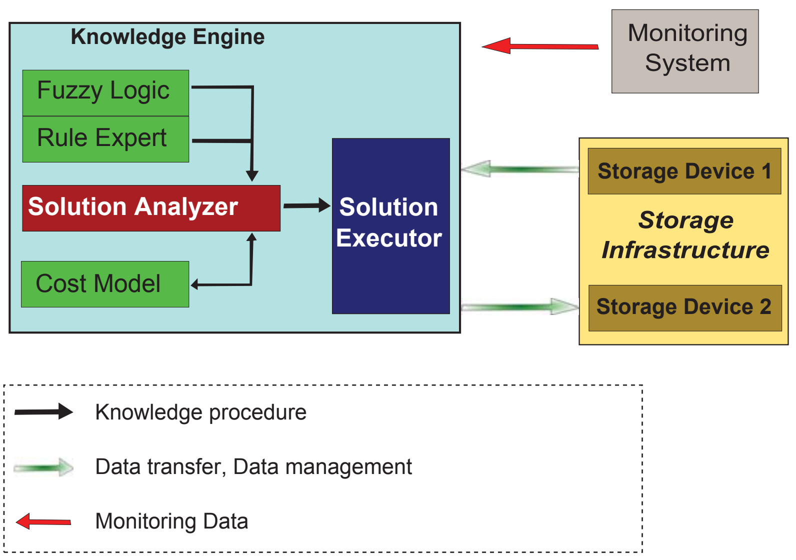


Figure 3. Communication between modules when optimizing data location.

In this case the system is able to relocate replica, create a new one and delete those ones which are not necessary anymore. All these actions are performed and selected based on the possessed knowledge and the available cost model.

5. Tests and performance issues

Now, we are going to present tests which were performed on our test infrastructure. We present two tests which were used to verify the usability of our knowledge engines and our approach for the data storage management.

In the first test we used only a single host on which the whole system has been installed. In this case when the system doe not have any possibilities for optimization it would behave like a simple FTP server. But the time which is needed for download/upload operations was much higher than with the FTP. It is caused by the knowledge engines which were more time consuming.

The second test was more complex. It was performed on a infrastructure which contains four hosts presented in Table 1. Three of them have storage capabilities (two possess an FTP server and one has a storage site library – SiteManager which was connected to the knowledge engine). The knowledge of this system and the main part which is responsible for the optimization of usage of the infrastructure were located on a fourth host. This host does not have any storage capabilities and was used as the central point which deals with the user requests and is able to manage the whole infrastructure.

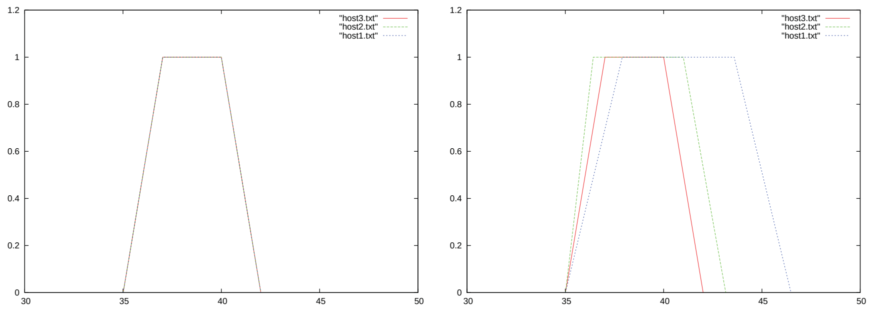
Table 1
Hosts used in the tests performed.

Host	Operating System	FTP Server	SiteManager	Knowledge Engine
1	Ubuntu Linux 10.04	+	–	–
2	Ubuntu Linux 10.10	–	+	–
3	Ubuntu Linux 10.04	+	–	–
4	Ubuntu Linux 10.04	–	–	+

During this test we use a knowledge engine which is based on fuzzy logic represented by fuzzy sets. In the test we want to observe changes to the knowledge described by the sets which refer to individual actions. We performed 43 similar events, where 27 of them were assigned to the action described for host1, 13 of them were assigned to host2 and the rest (3 events) were assigned to host3. Due to this set of events the size of the fuzzy sets which represents the knowledge was changed. The set which refers to host1 was expanded by about 42%, the set which refers to host2 was expanded by about 12,5%. The third set was expanded by ca 0,5%. This situation is presented in Figure 4 where we illustrate one of our tests which was performed in the second scenario and uses fuzzy logic as a source of the knowledge engine. In this example we present only one of the dimensions of our sets which was related to the read operation speed of each host. In this test the user wants to achieve the best read speed for his/her data. In this example the dimension possesses only three sets which describe the first three hosts presented in Table 1. During the tests, host one was preferred and host three was selected occasionally. After a few iterations the set related to host 1 was expanded (blue color). The set of host 2 was expanded to a little extent (green color). The last set referring to host 3 was the same at the end as at the beginning (red color). The modification of each fuzzy set is important because the wider sets can provide a better solution for a wider range of parameters under monitoring. By modifying these sets the system tries to adapt its knowledge to the observed situation. In the tests under discussion this modification allows us to prefer the storage device which is able to offer the best read speed (this host has the widest set).

During this test we observed that our system was able to adapt to the monitored situation. The data which were added to the storage infrastructure were distributed among individual storage locations. This distribution was aimed to meet the user requirements as to the read operation speed. We also observed that not always the location of the data was selected to optimize the resource usage. Nevertheless, the selected location allows for meeting the user requirements.

During the performed tests we can observe that the usage of knowledge engine which is based on rules takes some time. For example for small tests which comprise only a few rules (up to five) the analysis of the knowledge may take a few seconds.



(a) Initial state (sets are similar) (b) State after knowledge modification

Figure 4. Fuzzy logic sets for each host, before and after testing.

For bigger knowledge, an analysis may take much longer even up to a half a minute. During tests we observed that this is caused mainly by the time which is needed to prepare and run the Drools-based knowledge engine. The second kind of knowledge engine which is based on fuzzy logic is much faster than the previous one. The time of knowledge analysis by this engine is dependent on the number of measured parameters and on the number of sets which describe the possible solutions. This engine is quite time effective for our needs. The engine which is based on rules is not as good in the current implementation but we plan to optimize its work and reduce the time consumed to prepare the knowledge for analysis, but at the current level it is quite satisfactory for our needs, as well as for our tests. In other words, we aim at finding a reasonable compromise between the costs and gains of the AI usage.

6. Conclusions and future work

In this paper we present a system which is able to manage data storage devices. Our system uses two methods of artificial intelligence they are fuzzy logic and rules. Both methods are used to find a solution (sequence of actions) when the infrastructure under monitoring can be optimized. Our system is responsible for replica management and data location in the infrastructure. This approach exploits not only AI methods but also it introduces the cost model which is used to pick the cheapest action from the actions proposed by the knowledge engines. Our system is developed to offer scientists the storage infrastructure which is able to fulfill their requirements relating to data storage with the minimum cost. This approach is designed for distributed data storage. During tests we stated that our system was able to adopt to the observed situation in the monitored infrastructure. Also, it is able to fulfill user requirements and minimize the cost of storage.

In the future we are going to add a new functionality to our system which will be connected with context search when an action of our system was performed – it is aimed to distinguish between possible contributions to performance improvement.

This functionality will allow for better fitting of the system responses to the infrastructure behavior and the current data storage state.

The second function which we want to extend our system by is a database matching which will try to find a historically performed action and match it to the observed situation. The database matching will consist of a few steps. In the first one the system seeks the database and tries to match the fields of the database with the obtained parameters from the monitoring system. If most of the obtained parameters are identical or similar to one of the database records the system performs the action which is stored in the database for the found record which is similar to the observed situation. The comparison of the records and obtained parameters will be parameterized and not all the parameters have to be the same or even measured/stored. The number of the stored and monitored parameters may differ but most of them should be taken into account to determine if the stored action may be used. The aim of this function is to reduce the overhead induced by the actions which are often performed. The database is going to be analyzed with the DM methods.

A further functionality which we want to add in the nearest future is the ability to define and analyze the behavior of the user. This function is aimed to determine relationships between user requirements, data usage and the state of the monitored infrastructure.

Acknowledgements

This research is partly supported by the European Union within the European Regional Development Fund program as part of the PLGrid PLUS Project POIG.02.03.00-00-096/10 (`http://plgrid.pl/plus`) *and the AGH grant 11.11.230.015.*

References

[1] Drools expert user guide. `http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html/index.html`.

[2] Nagios core 3.x documentation. `http://nagios.sourceforge.net/docs/nagioscore/3/en/toc.html`.

[3] Zabbix 1.8 documentation. `http://www.zabbix.com/documentation/1.8/manual`.

[4] Abdurrab A. R., Xie T.: FIRE: A file reunion based data replication strategy for data grids. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 215–223, May 2010.

[5] Ben Charrada F., Ounelli H., Chettaoui H.: An efficient replication strategy for dynamic data grids. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, pp. 50–54, Nov. 2010.

[6] de Franceschi A., da Rocha M., Weber H., Westphall C.: Proactive network management using remote monitoring and artificial intelligence techniques. *Com-*

puters and Communications, 1997. Proceedings., Second IEEE Symposium on, pp. 167–171, July 1997.

[7] Funika W., Korcyl K., Pieczykolan J., Skital L., Balos K., Slota R., Guzy K., Dutka L., Kitowski J., Zielinski K.: Adapting a hep application for running on the grid. *Computing and Informatics*, vol. 28, pp. 353–367, 2009.

[8] Hellmann M.: Fuzzy logic introduction. `http://diuf.unifr.ch/ds/courses/dss2002/pdf/FuzzyLogic.pdf`.

[9] Krol D., Funika W., Slota R., Kitowski J.: SLA-oriented semi-automatic management of data storage and applications in distributed environment. *Computer Science – Annual AGH-UST*, vol. 11, pp. 37–50, 2010.

[10] Krol D., Kryza B., Skalkowski K., Nikolow D., Slota R., Kitowski J.: QoS provisioning for data-oriented applications in pl-grid. *Cracow Grid Workshop – CGW'10*, pp. 142–150. ACC-Cyfronet AGH, October 2010.

[11] Lamehamedi H., Shentu Z., Szymanski B., Deelman E.: Simulation of dynamic data replication strategies in data grids. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 75–86. IEEE CS Press, April 2003.

[12] Prodan R., Nae V.: Prediction-based real-time resource provisioning for massively multiplayer online games. *Future Generation Computer Systems*, 25(7): 785–793, 2009.

[13] Ranganathan K., Foster I. T.: Identifying dynamic replication strategies for a high-performance data grid. In *Proceedings of the Second International Workshop on Grid Computing*, GRID '01, pp. 75–86, London, UK, UK, 2001. Springer-Verlag.

[14] Ribler R. L., Simitci H., Reed D. A.: The autopilot performance-directed adaptive control system. *Future Gener. Comput. Syst.*, 18(1):175–187, September 2001.

[15] Sota R., Krl D., Skakowski K., Orzechowski M., Nikolow D., Kryza B., Wrzeszcz M., Kitowski J.: A toolkit for storage qos provisioning for data-intensive applications. *Computer Science*, vol. 13 (1): 63–73, 2012.

[16] Wang X., Yang G., Li Y., Liu D.: Review on the application of artificial intelligence in antivirus detection systems. In *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, pp. 506–509, September 2008.

[17] Wismueller R., Bubak M., Funika W., Arodz T., Kurdziel M.: Support for user-defined metrics in the online performance analysis tool G-PM. In Dikaiakos M. D., editor, *Grid Computing European Across Grids Conference AxGrids'04*, volume 3165 of *Lecture Notes in Computer Science*, pp. 159–168. Springer, January 2004.

Affiliations

Włodzimierz Funika
AGH University of Science and Technology, Krakow, Poland, `funika@agh.edu.pl`

Filip Szura
AGH University of Science and Technology, ACC CYFRONET AGH, Krakow, Poland,
`szura@agh.edu.pl`

Received: 30.08.2012
Revised: 23.10.2012
Accepted: 11.02.2013