Automation of distributed data management in applied microservices package for scientific computations

G A Oparin, V G Bogdanova and A A Pashinin

Matrosov Institute for System Dynamics and Control Theory of SB RAS, Lermontov St. 134, Irkutsk, Russia, 664033

bvg@icc.ru

Abstract. We offer a specialized toolkit for automating both knowledge management when creating an applied microservices package and data accumulating during its application for scientific computations in a hybrid computing environment. The decentralized solving of the declaratively formulated problem is carried out by an active agent group. This group is self-organized by logical inference on the distributed knowledge base of a subject domain. The developed toolkit automates the creating and updating of the local knowledge base of the manager-agent of applied microservices package, as well as the local knowledge bases of distributed computational agents. Local knowledge bases are formed using a description of the interface of computational microservices managed by these agents. Microservice ensembles, corresponding to the active group, are stored in the knowledge base of the manager-agent. The developed toolkit uses this information for testing microservice in the case of its update. In hybrid computing, this toolkit provides synchronizing, archiving, and saving of calculated data. Hybrid infrastructure combines the reliability and availability of using on-premises computers with scaling to the cloud when peak loads occur. The conducted experiments confirmed the effectiveness of the presented approach for solving practically significant scientific problems.

1. Introduction

The Applied Microservice Packages (AMP) [1] provides an effective way for distributed scientific computations. Intelligent AMPs are based on a subject domain (SD) model, which is understood as a collection of information about SD objects and the relationships between them. The user is allowed to formulate the problem of finding the desired quantities by given initial values (non-procedural problem statement, NPS) without previously studying the relationship of applied modules using AMP agents. These agents deployed in a specific computational field (CF) [1, 2] launch the microservices implementing applied modules. AMP agents manage by solving the user's problem. Management is decentralized for NPS and performed by an active group of agents based on pair interactions. This group self-organizes by the logical inference using a knowledge base (KB) distributed over CF nodes. Fragments of relations are stored in local KB of agents, reflecting the relationship of applied modules with input and output parameters. This knowledge identifies neighboring agents involved in pair interactions. The event management by input data readiness coordinates the behavior of agents both in the formation of a group and performing joint actions for solving the user problem.

Agents are equivalent in the sense that they have the same behavioral model. Thus, agents deployed on nodes of a CF are a computational system consisting of a set of equivalent nodes that jointly perform a common problem and are connected with neighbors through links to a logical network. According to the definition given in [3], in the most general sense, this network of computational agents is a peer-to-peer overlay network due to its properties. Combining computational agents into a network, the nodes

of which interact in the P2P computing style [4], provides the ability to create open multi-agent systems (MAS). The set of functioning agents is dynamical in such MAS [4]. Thus integration a reliable and scalable P2P system with multi-agent technology provides advantages of both paradigms.
The use of agent architecture jointly with microservice one provides a practical opportunity to implement the mechanisms of semantic interaction of agents in a P2P network. The HPCATAMP technology, developed by authors, for creating and using AMP, in which the rights to execute microservices are delegated to computer network agents, is described in [1]. The new instrumental basis of this technology is the cloud version HPCSOMAS-MSC of software platform HPCSOMAS-MS [1]. In this study, we offer AMPDM (AMP Data Management) tools implemented in this version for automation both knowledge management during the creating of AMP and saving calculated data during its application. The AMPDM is oriented to automate the AMP usage for solving problems in a hybrid computing environment. Hybrid cloud infrastructure extends the local user infrastructure with cloud resources in case of peak loads [5]. The part of the cloud infrastructure hosted at the client resources on which cloud services are installed is named the on-premises cloud infrastructure. Developed tools are used in the AMP deployed in a hybrid infrastructure. AMPDM automates the creation and updating of the local KB of the AMP manager-agent (AMPMA), as well as the local KB of the distributed computational agent (DCA). Local KBs of DCAs are formed based on a description of the interface of computational microservices managed by these agents. Information about active DCA groups, according to NPS, are stored in KB of the AMPMA for using in the process of testing updated microservices. In the process of hybrid computing, AMPDM provides synchronization, archiving, and storage of calculated data.
A microservice-based approach, a hybrid environment, and synchronization of data installed on both cloud resources and on-premises computers make our technology closer to the dew computing (DC) technology. DC oriented to the full use of the potential of on-premises resources and scalability of the cloud [7]. DC is based on the concept of microservice in a vertical hierarchy of distributed computing [8]. The DC categories underlined in [9] by the resource used within this paradigm, are supplemented by a new one – AMP in Dew. In this case, on-premises computer resources such as agents, microservices, and a user-selected data set (in a synchronized folder) have a cloud copy. The identity of these resources is maintained using AMPDM. The proposed approach is demonstrated by the example of the AMP developed with its use for solving problems of the qualitative study of binary dynamic systems (BDS) based on the Boolean constraint method (BCM [10]). The relevance of research is due to the widespread use of BDS as models of gene regulatory networks in bioinformatics. The conducted experiments confirmed the effectiveness of the presented approach to solving practically significant scientific problems.

2. Related work

The creation and application of peer-to-peer computing networks [11] and also the study of methods of its construction are actively developing in recent years [12]. Computer services are moving from traditional Internet services to a peer-to-peer cloud (P2P Cloud), which is reflected in a wide range of scientific studies. The integration of cloud computing and P2P systems is used to overcome the problems that arise in these paradigms. The advantages of P2P networks, discussed in detail in [3, 11, 12], are scalability, communication costs minimization, self-organization, and adaptability. Distribution and decentralization of such networks provide fault tolerance and load self-balancing [13]. In [12], the study of the P2P networks application is given, in particular, their use for parallel computing and the creation of fault-tolerant systems was noted. One of the problems in cloud computing is the use of centralized approaches for servicing discovery, monitoring, and load balancing. Therefore, many decentralized mechanisms for storing data in the cloud are based on P2P protocols [14]. Currently, most P2P systems operate with a simple content exchange, which is significantly different from the functionality of the services exchange. A review of several works devoted to the development of research related to the integration of P2P networks and cloud computing (for example, [15-17]) to support such functionality is given in [14]. An efficient and scalable P2P Cloud approach to providing cloud services using a structured P2P based on a distributed hash table was proposed in [15]. This approach is used for storage, discovery, and sharing of services.

In [16], a prototype of a P2P cloud system is proposed, namely, a fully distributed IaaS Cloud infrastructure. The authors note that, unlike the Volunteer Computing system, P2P Cloud does not have central coordination or central task repository. In [17], an approach is proposed to provide a highly accessible storage service in a mixed P2P Cloud. Mentioned above approaches [15-17] are oriented for the provision of storage services based on the P2P network, as noted in [14]. Although approaches, similar [15-17], are successfully used in the past two decades in integration with the cloud, studies related to its use as the basis for the provision of software as a service (SaaS) are still rare [14]. The following idea was proposed in [14], P2P Cloud approach (central server - primary administrator and set of nodes – super peers) provides remote executing of service in SaaS form hosted on peers.

Unlike the above works, our idea is to integrate the advantages of a P2P network into multi-agent technology to organize decentralized management of service-oriented scientific computing in AMP. The rights to execute computational microservices are delegated to agents. The problem solution is based on the semantic interaction of P2P network agents deployed on the nodes of the CF. Unlike [14], the hybrid cloud infrastructure is formed by the HPCSOMAS-MSC software platform based on a template similar to the "on-premises pattern" [6]. This pattern expands the scope of management tools by installing on the cloud resources special management agent of HPCSOMAS-MSC, CRA (Cloud Resource Agent). This approach requires the preliminary installation of additional software and DCA agents implemented as services under the capabilities of the resource and the class of problems. The installed agent controls computations only on its resources. The service-oriented computing paradigm contributes to the advancement of distributed systems and their applications significantly. Therefore, both tools for creating, deploying, and updating services are needed, as well as tools for automating data management during the development and application of AMP.

3. Data types in AMP

Scientific computations processes have unique requirements that are different from business work processes [18], for example, requirements arising from multivariate computations. Organizational requirements common to various SD include a user interface, reuse of computing components, end-user transparent runtime environment, support for local distributed computing, and pipeline-parallel computing, scalability, and fault tolerance. High-level management of both metadata and calculated data obtained during the computational experiments are required to fulfill all these requirements. The creation of AMP for scientific computing and the organization of the computing process is based on a multi-agent approach.

Figure 1 shows a diagram of the data exchange during the computation process in AMP. Compared to earlier studies [19], the management of computations is simplified. The top-level of control is comprised of AMPMA. These agents are the advanced version of the previously developed Problem Statement Agent (PSA, [20]). Along with providing an interface for NPS, these agents have significantly extended functionality, in particular, the data management services discussed in the next section. DCAs are at a lower level of management. The DCA is the modification of Distributed Solver Agents (DSA [20]) by including the functionality of running Computational Microservice (CM), which was previously the prerogative of individual computing agents.

The computational process begins with the NPS in the user interface provided by the AMPMA agent. A Dew user within the DC paradigm performs NPS on a Dew-AMPMA agent installed on an on-premises resource. NPS input and output parameter names are selected from the AMPMA agent parameter vocabulary. These vocabularies may vary depending on the specific AMPMA. The steps for solving the problem are described in detail in [20]. At the first stage, the conclusion about the problem solving possibility is made based on logical inference on the DCA agent KB distributed across the nodes of the CF. This KB stores, in particular, fragments of interconnections of parameters and computational modules. The NPS and the corresponding active group of agents are stored in the AMPMA local KB. This data is duplicated on other AMPMA to provide fault tolerance computing. At the stage of joint actions, only agents of the active group operate, to which AMPMA transmits input data. The active group then performs a decentralized problem solving, regardless of the intervention of the AMPMA agent or user. This queue can arise both if the DCA is a member of different active

groups or take part in multivariate computations. If the reserves of the local infrastructure and the private cloud are exhausted, AMPMA calls the CRA in the case of the option of using an additional cloud resource was set in the user profile. DCA agents, analyzing task queue, decide on the need for the request to AMPMA and reaction in cases of both allocating the resources and request failure. Data management in the computation process is carried out by the AMPDM system.

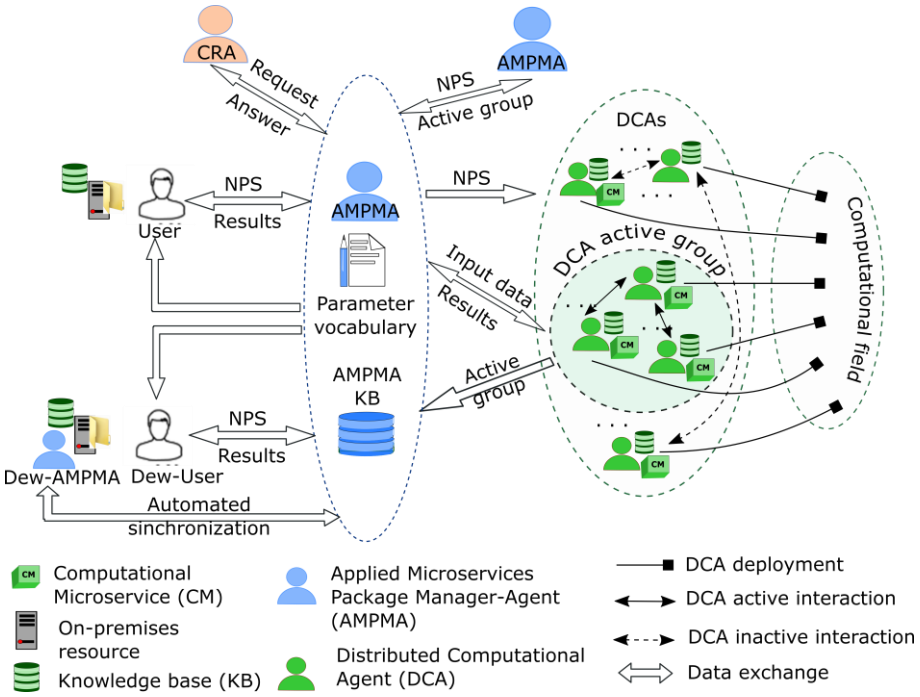

Figure 1. The scheme of the data exchange during computations.

4. AMPDM system design

AMPDM is intended for AMP developers, users, and administrators. Automation of data management using AMPDM is carried out both at the stage of development of AMP and the stage of its application, also in computational microservices updating. The composition of AMPDM is shown in figure 2. Automation of Data management is provided using the services of the AMPMA and several wizards of the MSCDT (Microservices Creation Deployment Testing, [6]) system:

- *Authorization manager* manages user authorization and the provision of access to agents, computing microservices, and services for viewing computation results.
- *File manager* manages the download, editing, and backup copying of user files by AMPMA agents. Data exchange is performed using the HTTP protocol.
- *Synchronization manager* manages the synchronization of local KB, calculated user data of DSA agents, and NPS formulated on AMPMA.
- *Creation wizard* automates the creation of new services based on ready-made standard templates and filling its local KB.
- *Configuration wizard* automates the setup of agent parameters, the organization of the relationship between computing agents and microservice functionality.
- *Test wizard* automates the testing of a deployed package and the interaction of agents and microservices. The user can send test tasks to microservices or their ensembles and check the result.
- *Update wizard* automates updating agents and microservices from local and external repositories.

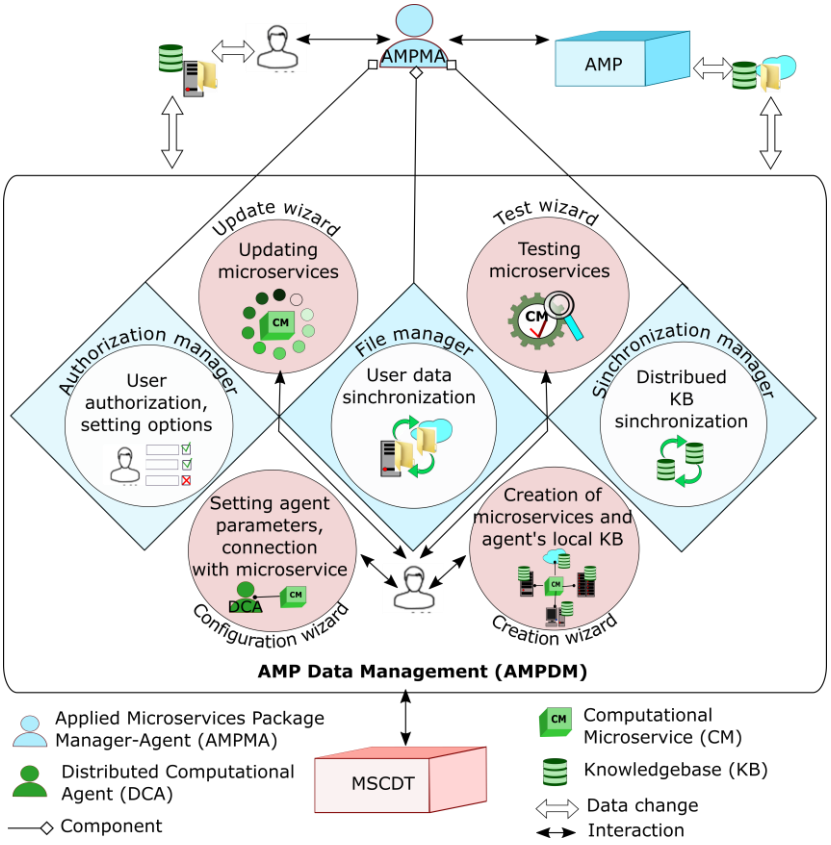

Figure 2. The composition of the AMPDM.

Managers are accessed through the AMPMA web interface.

4.1. Backup copying and synchronization

Backup copying of the results is needed in the following cases:

- The results take up much space, and their long-time storage on a computing resource is inappropriate.
- The results are significant, and there is a danger of their loss when stored on a single computing resource.
- Computations are carried out on the temporarily allocated resource, which must be disabled after experiments.
- Regular work is required with these results on the user resource.

The user can install the Dew-AMPMA on his computer and configure it to synchronize the results of computations and input files from the available AMPMA agents, by request, schedule, or the fact of data change. A user who has direct access to two or more AMPMA agents at once can configure them to back up or synchronize the computation results from one to the other, and mutually between them. The administrator can also configure AMPMAs so that user files are moved to agents available to these users, where data storage will be more efficient. Synchronization in the computation process is carried out for the following objects:

- Files uploaded by the user,
- Files created as a result of computations,
- Database, in which information is stored about completed tasks, including the NPS, its launch time and data, active group, and other information needed later.

Figure 3 shows the user profile, synchronization, and backup settings performing by the file manager of the external (not located on the user's local computer) AMPMA. The interface is designed to set the synchronization mode, by command or by time, indicating the frequency of synchronization, for downloading data, changing the password, mail, and for subscribing to notifications.

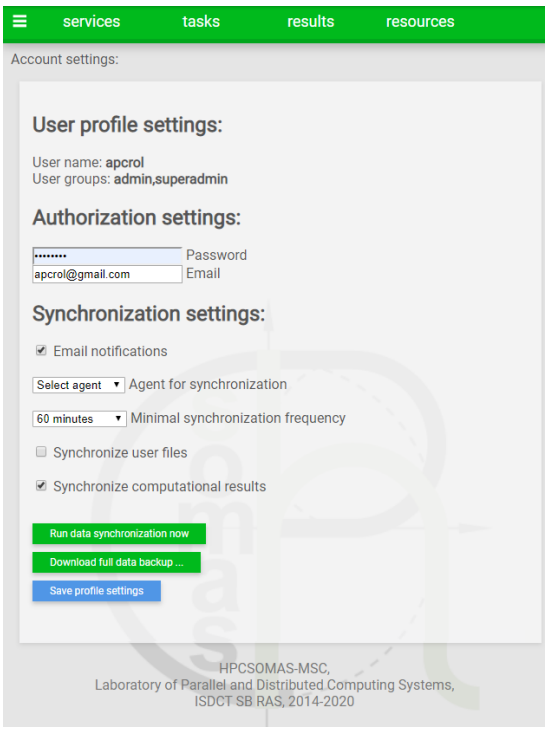

Figure 3. APMA interface for user settings.

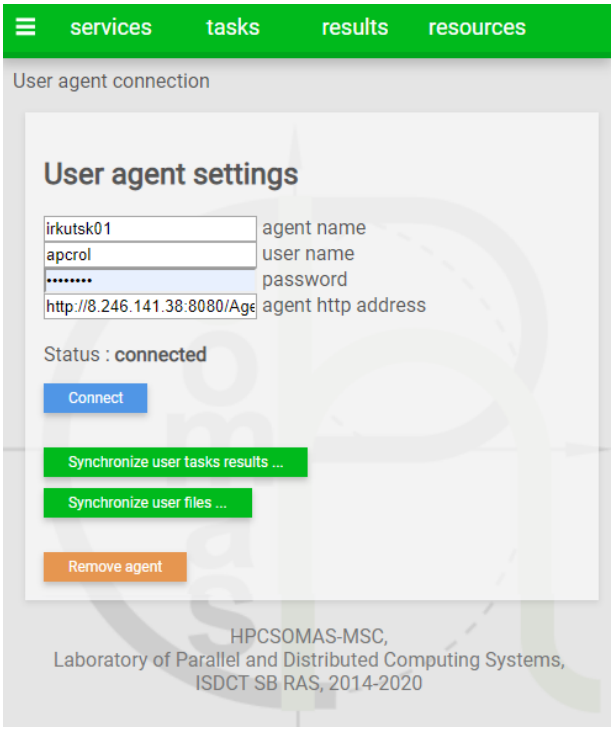

Figure 4. Dew-APMA interface.

DC paradigm based data synchronization is used when restoring communication after the Internet disconnecting. This synchronization is performed between the user's Dew-AMPMA and the specified external AMPMA. The synchronized data search is based on the file modification time stored in the update table of APMA.

The Dew-AMPMA must be pre-connected to external AMPMA (figure 4). This connection provides the ability to exchange data, requests for solving the problem through Dew-AMPMA, synchronize files between these agents, and back up the computation results.

4.2. Updating and testing of microservices

To update computational microservices, the MSCDT system update wizard is used. The update is performed in case of changing the application module, based on which the microservice is implemented. As a result, the microservice interface may remain the same or change. In the first case, the update wizard calls the AMPMA agents on which the microservice is registered. Each AMPMA agent checks whether the microservice is currently activated as part of active groups of agents that perform computations on the user's current request. If the microservice is involved in the computation process, AMPMA blocks updates until it is completed. Then AMPMA puts the microservice into an unavailable state, and the update wizard replaces it in the AMPMA repository. In the second case, modification of the distributed AMP KB is additionally required. User request for NPS is blocked until the update and testing are done.

Autonomous testing of the functionality of the updated microservice and then end-to-end testing [22] of the AMP is carried out using the testing wizard. The end-to-end testing is aimed at verifying the correctness of solving problems involving this microservice by the test NPS and data stored in the

system, which provided by the developer. During end-to-end testing with the presence of test NPS in the system, the test wizard automates the launch of these NPS and verifies the results of this launch. Test wizard automates the processing of the following situations:

- The test results are correct, but the composition of the active agent group has changed. This composition for the tested NPS is updated after the user confirms this action.
- Test results are incorrect. The user receives a list of NPS given these results.
- The results are not obtained, because, during the assembly of the active group, the agent managing the updated microservice did not wait for the input data. This situation arises when the interface of the updated microservice changes in such a way that it makes it necessary to include some of its input parameters in the NPS. The user receives a list of such incomplete NPS.

Conducting end-to-end testing in semi-automatic mode (in the absence of test NPS, for example, when creating AMP), the user performs the request for the NPS in the APMA interface. Upon receiving the correct results, AMPMA saves the test NPS and data in its KB.

5. The example of using the AMP

The developed tools are tested on the example of creating and using AMP for a qualitative study of BDS based on the Boolean constraint method [10]. All computing microservices implemented based on this approach can be divided into the following groups:

- Microservices for creating Boolean models,
- Microservices for checking the feasibility of Boolean constraints (SAT and 2QBF solvers),
- Microservices for pre- and post-processor processing.

Microservices for creating Boolean models implement the following capabilities:

- Construction of the function of a one-step transition Φ_1 according to the description of the BDS dynamics equations,
- Construction of the *k*-step transition function Φ_k based on the function Φ_1,
- Construction of the model for verifying the presence of a specific dynamic property in the BDS based on the function Φ_1 or function Φ_k, and the property specification.

The correct functioning of these microservices is verified by the satisfiability checking of the obtained Boolean models of the dynamical properties for a specific previously studied BDS by SAT or 2QBF solvers. If obtained on the model basis conclusions about the presence of verified dynamic properties coincide with the results obtained earlier for the same BDS, then the model is constructed correctly. In the same way, models for synthesizing the control laws of dynamic objects are tested. As a test for the AMPDM system, BDS of small dimension is chosen to reduce the time for subsequent testing when updating microservices.

A set of BDS for automated testing in AMPDM was formed in the process of developing computing services for building Boolean models for a qualitative study of BDS and the laws of their management. For example, classical examples of gene regulatory networks [22] were used for searching the equilibrium states [23]. An example from [24] was used for studying the properties of reachability type [10]. The results of constructing attractor basins were compared with [25], the model for searching for cycles of the given lengths were tested using examples from [26, 27]. The BDS from [28] was used for testing the synthesizing of the stabilization output feedback [29]. Stability properties when analyzing the state space of the shift register were considered in [19] using the example from [30].

Let us consider the problem of constructing control sequences for BDS with the following dynamics:

$$x^t = F(x^{t-1}, u^{t-1}), \quad (1)$$

where $x \in B^n$ is the state vector, $B = \{0,1\}$, $u \in B^m$ is the input control vector, n and m are correspondently dimensions of state and control vectors, $t \in T = \{1,2,...,k\}$ is discrete time (tact number), $F(x, u)$ is a vector function of logic algebra, called the transition function ($F : B^n \times B^m \to B^n$). For each state $x^0 \in B^n$ and finite sequence $u(t) = (u^0, u^1, ..., u^{k-1})$ of states of the control vector u ($u^t \in B^m, t \in T$), we define the trajectory $x(t, x^0, u(t))$ of the system (1) as a finite sequence of states $x^0, x^1, ..., x^k$ from the set B^n. The problem statement is as follows. Let two states be given $c^0, c^* \in B^n$, where c^0 is the initial state, and c^* is the final (target) state. It is necessary to synthesize such a control $u(t)$ (if it exists) that transfers system (1) from the initial state c^0 to the final state c^* in k time steps.

In [20], the authors showed that the necessary control $u(t) = (u^0, u^1, ..., u^{k-1})$ (if it exists) is a solution of the Boolean equation

$$\Phi_k(x^0, x^1, ..., x^k, u^0, u^1, ..., u^{k-1})\big|_{\substack{x^0 = c^0 \\ x^k = x^*}} = 0. \quad (2)$$

Let us represent the conditions $x^0 = c^0$ and $x^k = c^*$ in the form of Boolean equations. Then (2) takes the form

$$\vee_{i=1}^{n} (x_i^0 \wedge \overline{c_i^0} \vee \overline{x_i^0} \wedge c_i^0) \vee \vee_{i=1}^{n} (x_i^k \wedge \overline{c_i^*} \vee \overline{x_i^k} \wedge c_i^*) \vee \Phi_k(x^0, x^1, ..., x^k, u^0, u^1, ..., u^{k-1}) = 0,$$

where the subscript i denotes the component number of the vectors x^0, c^0, x^k, c^*.

This problem is currently one of the most important [31] for studying the dynamics of the behavior of gene regulatory networks represented by a model discrete in time and state (1). From a theoretical point of view, this model is essentially a nonlinear model, and existing methods of control theory do not apply to it [20]. From a practical point of view, the resulting control may be useful for creating medications [32].

Let us find control sequences of length k=3 for BDS, the dynamics of which is represented by the following equations [32]:

$$x_1^t = u_1^{t-1}, x_2^t = x_1^{t-1} \wedge u_2^{t-1}, x_3^t = x_2^{t-1} \wedge x_5^{t-1}, x_4^t = x_3^{t-1}, x_5^t = \overline{u_3^{t-1}}, x_6^t = x_3^{t-1} \vee x_5^{t-1}. \quad (3)$$

The Boolean model is built for $c^0 = (101100)$ and $c^* = (011001)$ in the DIMACS format [33]. When constructing the function Φ_1 ($t \in T = \{1,0\}$), the encoding of the variables is as follows (table 1):

Table 1. Encoding of Boolean variables.

Variable	x_1^0	x_2^0	x_3^0	x_4^0	x_5^0	x_6^0	u_1^0	u_2^0	u_3^0	x_1^1	x_2^1	x_3^1	x_4^1	x_5^1	x_6^1
DIMACS format	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The function Φ_1 in the DIMACS format is represented in figure 5, where "-" is a logical negation, and "**0"** separates the disjuncts.

p cnf 18 15
10 -7 **0** -10 7 **0** 11 -1 -8 **0** -11 1 **0** -11 8 **0** 12 -2 -5 **0** -12 2 **0** -12 5 **0** 13 -3 **0** -13 3 **0** 14 9 **0** -14 -9 **0** 15 -3 -5 **0** -15 3 **0** -15 5 **0**

Figure 5. The function Φ_1 for (3) in the DIMACS format.

When constructing a function Φ_k based on Φ_1 for the next moment in time (with increasing t to k), the variable number during encoding increases by n+m. For k=3, substituting $x^0 = c^0$ and $x^k = c^*$ into

equation (2), we obtain for (3) the Boolean model for a search of the control sequence of length k=3 (figure 6).

```
p cnf 33 57
1 0 -2 0 3 0 4 0 -5 0 -6 0 -28 0 29 0 30 0 -31 0 -32 0 33 0 10 -7 0 -10 7 0 11 -1 -8 0 -11 1 0 -11 8 0 12 -
2 -5 0 -12 2 0 -12 5 0 13 -3 0 -13 3 0 14 9 0 -14 -9 0 15 -3 0 15 -5 0 -15 3 5 0 19 -16 0 -19 16 0 20 -10
-17 0 -20 10 0 -20 17 0 21 -11 -14 0 -21 11 0 -21 14 0 22 -12 0 -22 12 0 23 18 0 -23 -18 0 24 -12 0 24
-14 0 -24 12 14 0 28 -25 0 -28 25 0 29 -19 -26 0 -29 19 0 -29 26 0 30 -20 -23 0 -30 20 0 -30 23 0 31 -
21 0 -31 21 0 32 27 0 -32 -27 0 33 -21 0 33 -23 0 -33 21 23 0
```

Figure 6. The Boolean model for searching the control sequence of the length k=3.

After checking the satisfiability of this Boolean model using a computing microservice based on the AllSAT solver, we obtain three sets of solutions. Post-processing, which consists of choosing values from these solutions $u_1^0u_2^0u_3^0, u_1^1u_2^1u_3^1, u_1^2u_2^2u_3^2$, gives the following control sequences of length k=3:

$$\{101, 110, 011\}, \{111, 110, 011\}, \{100, 110, 011\}.$$

For the end-user, to construct the NPS for searching the control sequence (figure 7), it is necessary to set the input parameters: dimensions of the state and control vectors, the BDS dynamics description (3), length k, the initial state c^0 and final state c^* (respectively, parameters n, m, F, k, CI, CG in AMP). The control sequence or a set of control sequences (*CSLk* or *ACSLk* parameters in AMP, respectively) should be specified as the output parameter. In the first case, a computational microservice based on the SAT solver is called up to verify the model satisfiability; in the second case, the AllSAT solver is used for searching all solutions. The NPS based on proven models is added by the developer of a computing microservice to the AMPMA KB. When changing the microservices for constructing the function Φ_1 and Φ_k or constructing a Boolean model of the dynamic property, testing is carried out for all specified NPS using the testing wizard. Microservices for constructing functions Φ_1 and Φ_k are included in most NPS to solve the above problems. Therefore, testing automation significantly, by orders of magnitude, improves the microservice developer productivity in comparison with manually restarting NPS and verifying its results. After automated testing, a report is issued in which, for each running NPS, data are provided both for successful completion or failure in the execution process.

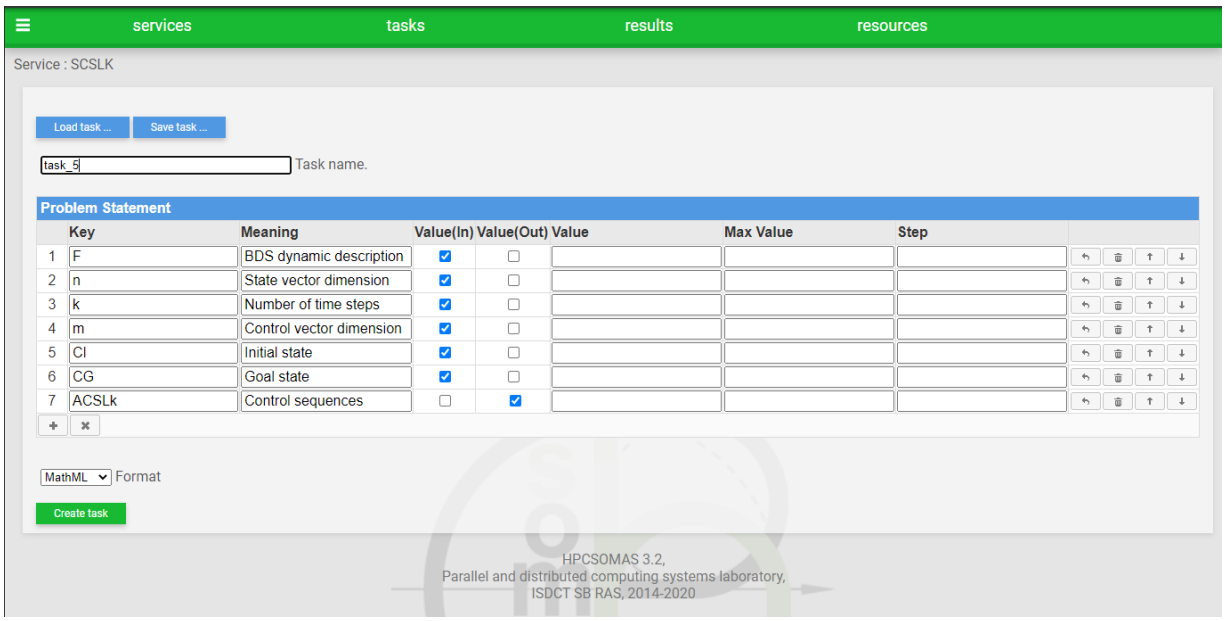

Figure 7. The NPS in the AMPMA interface for searching the control sequence of the length k.

6. Conclusion

The specialized toolkit is developed for automating knowledge management when creating and using a package of applied microservices and processing calculated data accumulating in the process of solving problems in a hybrid computational environment. The toolkit automates the creation and updating of the local knowledge base of both the agent-manager of the package of applied microservices and distributed computational agents. This toolkit provides various modes of data synchronizing between local and cloud resources, the updating and testing of computational microservices. The carried out experiments confirmed the effectiveness of the presented approach for solving practically significant scientific problems. The automating data management using this toolkit significantly improves the productivity of both the developer when creating and updating AMP, and the end-user of AMP in solving problems of qualitative research of BDS.

Acknowledgments

This work was supported by the Russian Foundation of Basic Research, project no. 18-07-00596. The authors would like to thank the Irkutsk Supercomputer Center of SB RAS for providing access to cluster computational resources.

References

[1] Oparin G A, Bogdanova V G, Pashinin A A and Gorsky S A 2019 Microservice-oriented approach to automation of distributed scientific computations *Proc. of 42st Int. Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019, Opatija, Croatia, May 2019* pp 253-258

[2] Viroli M, Damiani F and Beal J 2013 A calculus of computational fields *Advances in Service-Oriented and Cloud Computing (Comm. in Comp. and Info. Sci.* vol 393*)* ed C Canal and M Villari (Berlin, Heidelberg: Springer) chapter 10 pp 114-128

[3] Aberer K 2011 Peer-to-Peer Data Management *Synthesis Lectures on Data Management* **3**(2) 1

[4] Gorodetskii V I, Karsaev O V, Samoilov V V and Serebryakov S V 2008 Development tools for open agent networks *J. Comput. Syst. Sci. Int.* **47** 429-446

[5] Navale V, Bourne P E 2018 Cloud computing applications for biomedical science: A perspective *PLOS Comp Biol* **14**(6) e1006144

[6] CSCC: Practical Guide to Cloud Computing. [Online]. Available: https://www.omg.org/cloud/deliverables/CSCC-Practical-Guide-to-Hybrid-Cloud-Computing.pdf

[7] Skala K, Davidovic D, Afgan E, Sovic I and Sojat Z 2015 Scalable distributed computing hierarchy: cloud, fog and dew computing *Open Journal of Cloud Computing* **2**(1) 16-23

[8] Ray P P 2018 An introduction to dew computing: definition, concept and implications *IEEE Access* **6** 723-737

[9] Wang Y 2016 Definition and categorization of dew computing *OJCC* **3**(1) 1-7

[10] Oparin G, Bogdanova V and Pashinin A 2019 Qualitative analysis of autonomous synchronous binary dynamic systems *MESA* **10**(3) 407-419

[11] Buford J F and Yu H 2010 Peer-to-Peer Networking and Applications: Synopsis and Research Directions (*Handbook of Peer-to-Peer Networking* vol 34) ed X Shen and Yu H Buford et al. (Berlin: Springer Science and Business Media) pp 3-45

[12] Kryukov A P and Demichev A P 2018 Decentralized data storages: Technologies of construction *Program. Comput. Software* **44**(5) 303-315

[13] Daswani N, Garcia -Molina H and Yang B 2003 Open problems in data-sharing peer-to-peer systems *Proc. of the 9th International Conference on Database Theory (ICDT'03)* pp 1-15

[14] Achache A, Baaziz A and Sari T 2020 A hybrid P2P network-based remote treatment SaaS cloud computing *IJITEE* **9**(2S3) 575-583

[15] Zhygmanovskyi A and Yoshida N 2014 Cloud service provisioning based on Peer-to-Peer network for flexible service sharing and discovery *Journal of Computer and Communications* **2** 17-31

[16] Babaoglu O, Marzolla M and Tamburini M 2012 Design and implementation of a P2P cloud

system *Proc. of the 27th Annual ACM Symposium on Applied Computing* pp 412–417
[17] Kavalionak H and Montresor A 2012 P2P and Cloud: A marriage of convenience for replica management *Proc. of the 6th International Federation for Information Processing (IFIP) (Self-Organizing Systems. IWSOS 2012. LNCS* vol 7166*)* ed F A Kuipers and P E Heegaard (Berlin, Heidelberg : Springer) pp 60–71
[18] Klasky S, Beck M, Bhat V, Feibush E, Ludäscher B, Parashar M, Shoshani A, Silver D and Vouk M 2005 Data management on the fusion computational pipeline *J. Phys.: Conf. Ser.* vol 16 pp 510-520
[19] Bychkov I, Oparin G, Bogdanova V and Pashinin A 2019 Intellectual technology for computation control in the package of applied microservices *Proc. of the 1st Int. Workshop on Information, Computation, and Control Systems for Distributed Environments*, *Irkutsk, Russia, July* pp 15-28
[20] Oparin G A, Bogdanova V G, Pashinin A A and Gorsky S A 2018 Distributed solvers of applied problems based on microservices and agent networks *Proc. of the 41st Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* pp 1415-1420
[21] Newman S 2015 *Building Microservices* (O'Reilly)
[22] Dubrova E and Teslenko M 2011 A SAT-based algorithm for finding attractors in synchronous Boolean networks *Trans. Comput. Biol. Bioinformatics* (IEEE/ACM) **8**(5) 1393-1399
[23] Oparin G, Bogdanova V and Pashinin A 2019 Automation of microservices creation for qualitative analysis of binary dynamic systems *Proc. of the 1st Int. Workshop on Information, Computation, and Control Systems for Distributed Environments, Irkutsk, Russia, July* pp 88-98
[24] Vassilyev S N 2002 Attainability and connectedness in an automata network with a general state switching rule *Differ. Equ*. **38**(11) 1628
[25] Dubrova E 2008 Self-organization for fault-tolerance *Self-Organizing Systems. IWSOS 2008. LNCS* vol 5343) ed K A Hummel and J P G Sterbenz (Berlin, Heidelberg: Springer) pp145-156.
[26] Dubrova E and Teslenko M 2018 A SAT-based algorithm for finding short cycles in shift register based stream ciphers *Proc. of the IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2018* pp 65-72
[27] Oles V, Panchenko A and Smertenko A 2017 Modeling hormonal control of cambium proliferation *PLoS ONE* **12**(2) e0171927
[28] Li H and Wang Y 2013 Output feedback stabilization control design for Boolean control networks *Automatica* **49**(12) 3641
[29] Oparin G, Bogdanova V, Gorsky S and Pashinin A 2019 The synthesis of stabilizing feedback for binary dynamic systems: a logical approach *MESA* **10**(3) 477
[30] Massey J L and Li R W 1964 Application of Lyapunov's direct method to error-propagation effect convolutional code *IEEE Trans. IT* **10**(3) 248
[31] Akutsu T, Hayashida M and Tamura T 2008 Algorithms for inference, analysis and control of Boolean networks *Proc. of the Int. Conf. on Algebraic Biology* pp 1-15
[32] Moradi M, Goliaei S, Foroughmand-Araabi M-H 2019 A Boolean network control algorithm guided by forward dynamic programming *PLoS ONE* **14**(5) e0215449
[33] Satisfiability suggested format. [Online]. Available: http://beyondnp.org/static/media/uploads/docs/satformat.pdf