Research Article Open Access

Sergei Vostokin*, Yuriy Artamonov, and Daniil Tsarev

# Templet Web: the use of volunteer computing approach in PaaS-style cloud

https://doi.org/10.1515/eng-2018-0007
Received September 29, 2017; accepted November 22, 2017

**Abstract:** This article presents the Templet Web cloud service. The service is designed for high-performance scientific computing automation. The use of high-performance technology is specifically required by new fields of computational science such as data mining, artificial intelligence, machine learning, and others. Cloud technologies provide a significant cost reduction for high-performance scientific applications. The main objectives to achieve this cost reduction in the Templet Web service design are: (a) the implementation of "on-demand" access; (b) source code deployment management; (c) high-performance computing programs development automation. The distinctive feature of the service is the approach mainly used in the field of volunteer computing, when a person who has access to a computer system delegates his access rights to the requesting user. We developed an access procedure, algorithms, and software for utilization of free computational resources of the academic cluster system in line with the methods of volunteer computing. The Templet Web service has been in operation for five years. It has been successfully used for conducting laboratory workshops and solving research problems, some of which are considered in this article. The article also provides an overview of research directions related to service development.

**Keywords:** cloud computing, platform as a service, scientific computing, automatic parallel programming, cluster computing

***Corresponding Author: Sergei Vostokin:** Samara National Research University, Samara, Russia, E-mail: vostokin_sv@ssau.ru
**Yuriy Artamonov:** Samara National Research University, Samara, Russia, E-mail: ys.artamonov@yandex.ru
**Daniil Tsarev:** Samara National Research University, Samara, Russia, E-mail: daniil.tsaryov@gmail.com

## 1 Introduction

Modern mathematical modeling is based on numerical methods. Of all the numerical methods, the methods that are implemented exclusively in high-performance computing systems are becoming increasingly popular. Thus, the development of data mining or deep machine learning is fundamentally impossible without the use of high-performance technology.

Despite the affordability and widespread use of high-performance computing hardware, such as multi-core processors, general-purpose GPUs, and cluster-based systems, the developers of numerical models are facing problems in using such equipment efficiently. Firstly, it is important to access high-performance resources "on demand" in low-budget projects, preliminary studies, or training. Secondly, the terminal access over a secure channel based on SSH protocols requires developer to master the skills of system administration that are not typical professional skills for a numerical modeling expert. Finally, an important problem is the development of a parallel program for a numerical algorithm. The traditional form of representation of a numerical algorithm is a sequential program representation. But the modern tools for development of high-performance computing programs require an explicit description of simultaneously performed calculations and taking into account the hardware features of the computing equipment.

Modern mathematical modeling is used primarily for applied problems. Thus, reduction of the financial and time cost factors for setting up experiments with the models is practically important. In this regard, the motivation for our research in cloud computing service development is the problem of complex automation. The aim of automation is to reduce the negative role of the factors considered above. This automation includes: (a) the automation of the access procedure for a high-performance system; (b) the automation of program deployment (code upload, building, starting, monitoring the program execution, and downloading the results); (c) automatic parallel programming for high-performance computing systems. Development and maintenance of a service with the automation stated above are associated with solving a number of technical and scientific problems. Below we consider the methods developed and tested in the Templet Web service [1], statistical data on its functioning, and numerical modeling tasks solved with the help of it. In conclusion, the results of service development are summarized.

## 2 Methods

### 2.1 On-demand access to academic cluster

Our service is based on the idea of using volunteer computing to access a remote computer system. We define terms ‘Volunteer’ and ‘Consumer’ to explain how the on-demand access to an academic cluster is granted. A Volunteer is role name assigned to a person who has an account on a remote system accessible via the SSH protocol. A Consumer is role name assigned to a person who does not have an account on the remote system, but wants to use the Volunteer’s account. Running the Consumer’s program through the Volunteer’s account is implemented in our service using the following protocol (Figure 1):
- The Consumer submits the program source code and input data to Volunteer;
- The Volunteer runs the program on his/her own behalf on the remote system;
- The Volunteer returns result of the run to Consumer.

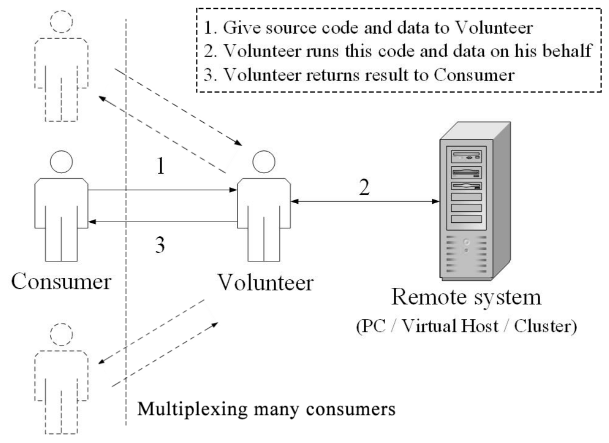

1. Give source code and data to Volunteer
2. Volunteer runs this code and data on his behalf
3. Volunteer returns result to Consumer
1
2
3
Consumer
Volunteer
Remote system
(PC / Virtual Host / Cluster)
Multiplexing many consumers

**Figure 1:** On-demand access scenario.

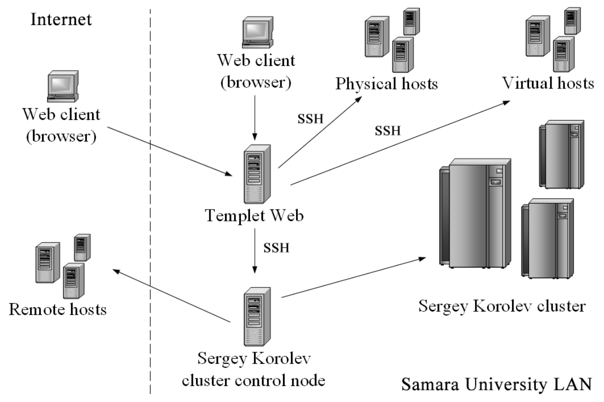


**Figure 2:** Computer environments available from the Templet Web system.

The system acts as a broker and assumes the following obligations:
- storing the source code, data, and the result of computations for mutual audit of the Consumer’s and the Volunteer’s actions;
- organizing the access to this information for both the Volunteer and the Consumer;
- multiplexing access for many Consumers to a single Volunteer account.

Each Consumer can interact with an arbitrary number of Volunteers; each Volunteer can simultaneously provide access to his remote system for an arbitrary number of Consumers. The roles of the Consumer and the Volunteer can be entitled to one and the same person.

Unlike the traditional BOINC [2] volunteer computing middleware, in our implementation a Volunteer grants access for a host with known IPs via SSH protocol, while BOINC uses agent-based approach for accessing a remote personal computer. The agent-based approach is preferable for PC that may be behind NAT or firewall. The SSH-based approach is preferable for cluster systems, for which Templet Web service was primarily designed. It is also assumed that the Volunteer gives access to multitasking systems (Figure 2).

Thus, if the Volunteer trusts the Consumer and knows his identifier in our system, he can immediately grant access to the Consumer.

### 2.2 Deployment automation

Three entities are involved in managing the deployment of user programs on a cluster: the Task, the Template, and the Environment.

The Task is the main entity of the system. The attributes of a task are the code of the Customer program, the input data, the output data and the execution status. The Task is generated from the Template.

Each Template contains sample code that users can adapt to their algorithm; the script that controls the assembly of the Task on the Volunteer system; there are also a start script and the script for downloading the results. Any Template can be used to create many Tasks. Each Task is related to one Template.

The Task is performed in an Environment. Each Environment contains information for connecting to the Volunteer system. The Environment implements the lifecycle of Tasks in the Volunteer system. Specific operations of the lifecycle are defined in the Template. For example, the Environment defines the moment when the program is started on the Volunteer system and runs the *build.sh* script specified in the Template.

The system has additional entities that are used to manage access rights and implement collaboration scenarios. The main scenarios include the work of a student group and the work of a research group on a cluster. You can grant access to one or more computing clusters to a group of students and monitor their work. The students can perform individual or group projects. The work on a project can be conducted using the browser or a version control system.

## 2.3 Automatic parallel programming

Let us demonstrate automatic parallel programming in our system using the example of a computation by master-workers scheme. This scheme is typical for volunteer computations in BOINC. When you create Tasks from the master-worker Template, the user is given the following code sample (some details are omitted).

```
struct task{
   /*-to be filled by the user-*/
};
struct result{
   /*-to be filled by the user-*/
};
struct bag{
  bool get(task*t){
   /*-to be filled by the user-*/
  }
  void put(result*r){
    /*-to be filled by the user-*/
  }
   /*-to be filled by the user-*/
};
void proc(task*t,result*r){
   /*-to be filled by the user-*/
}
int main(int argc, char* argv[])
{
    bag b;
    /*-to be filled by the user-*/
    b.run();
    /*-to be filled by the user-*/
    return EXIT_SUCCESS;
}
```

Here, *task* is the input data; *result* is the result of the calculation of the task; *proc* is the procedure for calculating the task performed by the workflow; structure *bag* is a state and methods of the master process; *get* is a method for creating a new task or informing that there are no tasks; *put* is a method for recording the result of calculating a task in a master process.

The user fills in the parts marked with the comment _/*-to be filled by the user -*/_, with his/her sequential code. The resulting skeleton can be compiled and checked for syntactical correctness in the usual way. The algorithm for converting the skeleton into executable code for a target system architecture is contained in Template. The system implements the conversion of this skeleton into the code to be executed in the shared memory using OpenMP, and into the code for distributed execution using MPI [3].

The pipeline skeleton is also implemented in Templet Web system. Besides, the system has a DSL-based constructor for developing skeletons in the form of actor networks.

## 2.4 Deployment scenarios

Automation of deployment and editing in the browser are the main steps of scientific application lifecycle in our service. Depending on the type of application, four deployment scenarios are supported.

The first scenario (Figure 3). In order to deploy classic BOINC applications on the academic cluster, we use the official addition to the BOINC system – the CluBORun package [4, 5]. In terms of our system, CluBORun is a special type of Environment (see Section 2.2; the other types of environments installed in the current version are a single workstation and a cluster running the TORQUE system). This type of deployment assumes that the BOINC manager is already configured and you need to specify its creden-

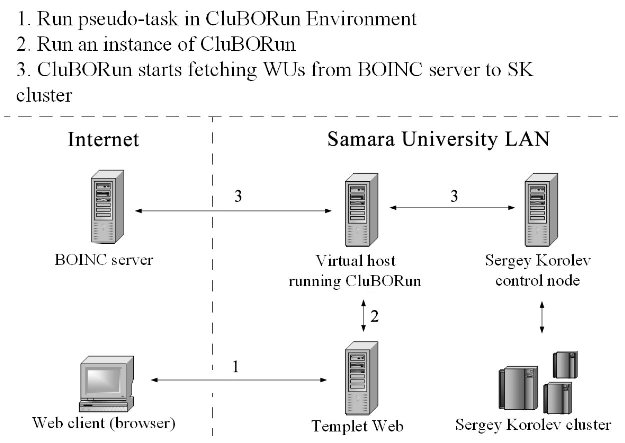

1. Run pseudo-task in CluBORun Environment
2. Run an instance of CluBORun
3. CluBORun starts fetching WUs from BOINC server to SK cluster
Internet
Samara University LAN
BOINC server
Virtual host running CluBORun
Sergey Korolev control node
Web client (browser)
Templet Web
Sergey Korolev cluster
1
2
3
3

**Figure 3:** Deployment scenario for the CluBORun.

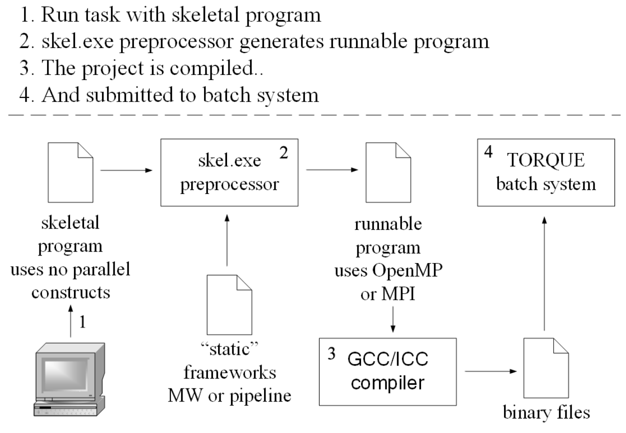

1. Run task with skeletal program
2. skel.exe preprocessor generates runnable program
3. The project is compiled..
4. And submitted to batch system
skeletal program uses no parallel constructs
skel.exe preprocessor 2
runnable program uses OpenMP or MPI
4 TORQUE batch system
"static" frameworks MW or pipeline
3 GCC/ICC compiler
binary files
1

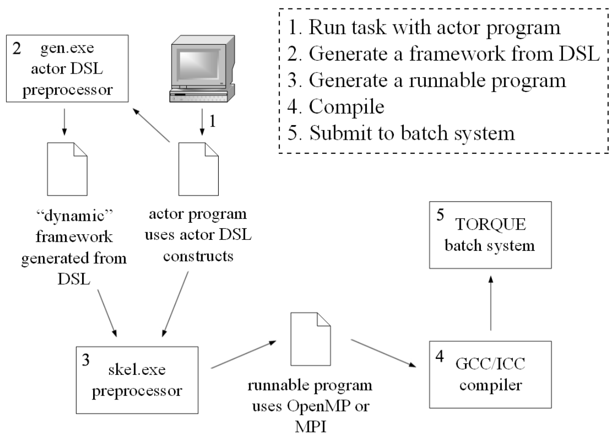
**Figure 4:** Deployment scenario for skeletal programs.

tials. The Volunteer account through which the cluster will receive BOINC work units (WU) using the CluBORun system is also specified. A CluBORun instance is launched on a dedicated virtual machine of our service. Thus, a Volunteer who has an account in the Templet Web system can process BOINC tasks using cluster resources in the automatic mode.

The second deployment scenario (Figure 4), also related to the BOINC concept, is the deployment scenario for skeletal programs with a fixed structure. A typical skeleton program for which this scenario is used is a program of the master-worker type, discussed in Section 2.3. This program can be used, for example, to prototype a BOINC application on a cluster. In order to deploy the skeleton program described in Section 2.3, a preprocessor is launched on the service side. It recognizes sections of code that the user has added to the skeleton instead of comments */ *-to be filled by the user - */*. Further, the recognized areas are automatically inserted into the corresponding places of the source code of the framework, which implements the skeleton program calculation model. The code generated this way is loaded onto the cluster management node, compiled, and launched in the batch system. This deployment scenario is more preferable and convenient for the user (compared to direct work with the framework) for the following reasons: (a) the system provides a ready-made skeleton and verifies its integrity during deployment, excluding programming errors; (b) complete encapsulation of the details of the framework: the user does not even know the names of its classes, methods, functions. Full concealment of implementation details allows you to apply different frameworks to deploy one skeleton program. For example, in the master-worker skeleton, different frameworks are now used to optimize computations in shared or distributed memory. In the future, we plan to extend this scenario for deploying master-worker applications directly to the BOINC control node.

The third scenario (Figure 5) assumes that the structure of the skeleton program is not static, but is defined using a domain-specific language (DSL). In our system, a DSL language is developed based on the concept of actors [6, 7]. The purpose of the language is to describe more complex flow control schemes, compared to the master-worker and pipeline. The structure of the skeleton in the third scenario is specified in the special *#pragma* directives. These directives are processed by the DSL preprocessor on the side of our service, resulting in the formation of the skeleton program. Further, similarly to the second scenario, the skeleton preprocessor checks the structure of the code. Then the recognized sections of the user code are automatically inserted into the corresponding places of the source code of our actor framework. Finally deployment is performed on the cluster management node in the form of an OpenMP or MPI application. In the future, it is planned to redirect the workflow generated by our actor framework to other external systems, for example, to BOINC or Everest [8].

The fourth type scenario is used in all other cases. This is the standard scenario for deploying programs on the clusters running TORQUE. The automation in this case lies in the fact that the user can apply the ready-made deployment scripts specified in the Template. In the most flexible case, with the permission of the Volunteer, who is the owner of the Environment, the Environment user is allowed to determine the necessary deployment steps manually in the *pbs* script.



1. Run task with actor program
2. Generate a framework from DSL
3. Generate a runnable program
4. Compile
5. Submit to batch system
2 gen.exe actor DSL preprocessor
1
"dynamic" framework generated from DSL
actor program uses actor DSL constructs
5 TORQUE batch system
3 skel.exe preprocessor
runnable program uses OpenMP or MPI
4 GCC/ICC compiler

**Figure 5:** Deployment scenario for actor programs.

## 3 Results

The Templet Web service is deployed in the private cloud of the Supercomputer Center at Samara University [9]. The service is used for teaching students high-performance programming, for automatic parallel programming technologies research, and for solving applied problems using numerical algorithms [10].

Most users are bachelor's and master's degree students. Students act as the Consumers of the service. Teachers who have accounts on the "Sergey Korolev" cluster act as the Volunteers providing resources for temporary access. The dynamics of user growth is shown in Table 1.

**Table 1:** The dynamics of Templet Web user growth.

| Year | Total in the period | Accumulated total |
|---|---|---|
| 2013-2015 | 212 | 212 |
| 2016 | 63 | 278 |
| 2017 (first six months) | 88 | 366 |

Table 2 shows the number of tasks running on the "Sergey Korolev" cluster during the period of operation of the Templet Web system. This number of tasks is multiplexed through three accounts on the cluster and one account on a test low-power Linux system.

**Table 2:** The number of task runs in the Templet Web system.

| Year | Total in the period | Accumulated total |
|---|---|---|
| 2013-2015 | 141 | 141 |
| 2016 | 2597 | 2738 |
| 2017 (first six months) | 1308 | 4046 |

Users develop tasks in projects. The project enables controlling access to Environments and Tasks for a group of users. Table 3 shows the dynamics of creating projects in the system.

**Table 3:** The dynamics of creating projects in the Templet Web system.

| Year | Total in the period | Accumulated total |
|---|---|---|
| 2013-2015 | 153 | 153 |
| 2016 | 126 | 279 |
| 2017 (first six months) | 237 | 516 |

In 2016, the function of code editing in a web browser was added to the system. As you can see from Tables 2, 3, 4, this function is in demand among users and has increased the intensity of Templet Web usage. Projects are basically created with the ability to edit the code in the browser.

**Table 4:** The number of Templet Web projects managed exclusively in the browser.

| Year | Total in the period | Accumulated total |
|---|---|---|
| 2013-2015 | 0 | 0 |
| 2016 | 97 | 97 |
| 2017 (first six months) | 226 | 323 |

In practice, the most commonly used deployment scenario is the standard scenario for deploying programs on the clusters running TORQUE. Deployment scenario for skeletal programs with a fixed structure is used in conducting parametric sweep calculations as described in [11]. The deployment of actor-oriented DSL programs is also implemented. The system includes a framework generator based on actor-oriented Templet markup language [6, 7]. The source code for the generator and some examples of how it is used in the deployment process are available online [12]. The CluBORun based deployment is currently in the testing phase.

An important part of the service is the subsystem for monitoring and forecasting the load of the "Sergey Korolev" cluster. It implements a 12-hour cluster load forecast, which is calculated by various mathematical methods: the maximum likelihood method [13], neural networks [14], an adaptive combination of listed methods [15]. The forecasting system is implemented using the microservice approach [16].

We use the Templet Web system to solve the problems of modeling the dynamics of space vehicles. The study of dynamics includes the numerical solution of spacecraft motion equations with different initial conditions, the construction of phase trajectories and Poincaré maps. The aim of the study is to identify chaotic processes and unstable modes in operation of the spacecraft orientation systems. This class of models is realized by the above-described scheme of master-worker calculations [11].

In this project, we also examined the applicability of the master-worker scheme to parallel algorithms implementation for training neural networks with the selection of the optimal structure of neurons in hidden layers [17]. A program for parallel continuous wavelet transform has been developed in the course of the study of the problem of analyzing acoustic signals from a cutting tool [18].

## 4 Discussion

The core of our strategy of providing access to a remote computing system is the concept of volunteer computing [19]. The donors of computing resources in our system are academic cluster account holders. The system is implemented as a PaaS (platform as a service) cloud service [20]. With this approach, the infrastructure that implements and provides the service is completely hidden from the user, and it is possible to work with the service using just a web browser.

The specifics of access to computing resources make us face the problem of forecasting the computational load of the cluster. A user who provides access to a cluster should know the periods when access can be granted without compromising his/her own projects and the overall cluster load. At the same time, a user who gets access needs to know when the efficient work on the cluster will be possible. In order to solve this problem, we use adaptive mixture of two forecasting models. The first one is extrapolation model on most similar pattern (EMMSP) in the history of a cluster load. The second one is multilayer perceptron with additional time-date factors. The perceptron extrapolates cluster load using the last available values of the load time series. The EMMSP model shows very good forecast on some time periods, but it is inferior to the perceptron in mean absolute error. The adaptive mixture of the two models shows the best mean absolute error, improving forecast of both EMMSP and perceptron models [15].

Parallel programming automation in our service is based on the concept of algorithmic skeletons [21, 22]. This method implies the storage of a set of frameworks for parallel computations management. The frameworks can be extended with sequential code for a task. In order to specify the semantics of parallel execution for algorithmic skeletons, we apply a variation of the actor model [6, 7] and define the model in the temporal logic of actions [23]. The syntax of skeletons is developed according to the language-oriented programming approach [24], considering the maximum compatibility with development environments and tools that are traditional for high-performance computing (C ++, OpenMP, MPI).

Many web-based systems for scientific application development were presented in the last decade. There are three major direction in development of the web-based systems. The first direction is focused on implementing remote access to scientific tools and computing resources via convenient web user interfaces [25]. The Templet Web system provides such an interface for utilization of free computational resources of an academic cluster in educational and research activities. The second direction is focused on creating and publishing complex workflow applications [26]. The third direction is API-centric approach that build platform around an open programming interface [27]. Following the latter approach the user can write a code in general-purpose programming language (for example in Python for the Everest web-service). This code looks like an ordinary single-threaded code for desktop computer, but it can potentially utilize huge computational power provided through the REST API. From this point of view, our system is similar to Everest. In contrast to Everest, we use the skeleton programming approach [21] to achieve the same functionality. That is why we use complex deployment which includes several code transformation stages. Such a deployment is performed transparently by the Templet Web service hiding complexity of underlying executing layers (currently C++ threads, Open MP and MPI). Our approach can also simulate workflows. We implemented a variant of actor model for developing workflow applications.

## 5 Conclusion

The experience we have gained from the development and operation of the Templet Web system shows the practical importance of a comprehensive approach to automating high-performance computing in mathematical modeling. This approach includes programming automation, on-demand access to resources and the deployment automation. The PaaS implementation of the Templet Web system is accessible through a standard web browser. This design significantly reduced the complexity of a cluster system for users and led to an increase of cluster usage. The technology of providing access to the computer system using the principles of volunteer computing has made it possible to simplify cluster administration considerably in the organization of the educational process using the resources of the Supercomputer Center of Samara University.

**Acknowledgement:** This work is partially supported by the Russian Foundation for Basic Research (RFBR#15-08-05934-A), and by the Ministry of Education and Science of the Russian Federation in the framework of the State Assignments program (#9.1616.2017/4.6).

## References

[1] Templet Web service home page: http://templet.ssau.ru/app

[2] Anderson D. P., Boinc: A system for public-resource computing and storage. In: Grid Computing, 2004 Proceedings, Fifth IEEE/ACM International Workshop on, IEEE, 2004, 4–10, DOI: 10.1109/GRID.2004.14

[3] Vostokin S.V., Tsarev D.A., Skeletal software deployment technology to automate the calculations on a supercomputer Sergey Korolev. In Advanced information technologies and scientific computing (PIT-2017): proceedings of the international scientific conference / ed. S.A. Prokhorov, Russia, Samara: Samara Scientific Center of RAS, 2017, 481-484 (in Russian)

[4] Afanasiev, A. P., Bychkov, I. V., Zaikin, O. S., Manzyuk, M. O., Posypkin, M. A., & Semenov, A. A. (2017). Concept of a multitask grid system with a flexible allocation of idle computational resources of supercomputers. Journal of Computer and Systems Sciences International, 56(4), 701-707

[5] Zaikin O., Manzyuk M., Kochemazov S., Bychkov I., Semenov A., A volunteer-computing-based grid architecture incorporating idle resources of computational clusters. In: International Conference on Numerical Analysis and Its Applications (2016 Jun 15), Springer, Cham, 2016, 769-776 (https://doi.org/10.1007/978-3-319-57099-0_89)

[6] Vostokin S.V., Templet: a markup language for concurrent actor-oriented programming, CEUR Workshop Proceedings, 2016; 1638: 460-468

[7] Vostokin S.V., The templet parallel computing system: specification, implementation, applications, Procedia Engineering, Vol 201, 2017, 684–689, DOI: 10.1016/j.proeng.2017.09.683

[8] Smirnov S., Sukhoroslov O., Volkov S., Integration and combined use of distributed computing resources with Everest, Procedia Computer Science, Vol 101, 2016, 359-368, DOI: 10.1016/j.procs.2016.11.042

[9] Supercomputing Center of Samara University, Templet Web manual: http://hpc.ssau.ru/node/3130 (In Russian)

[10] Artamonov Y.S., Vostokin S.V., The use of cloud services Templet Web in conducting laboratory workshops on the supercomputer "Sergey Korolev", In: Proc. of X Int. Scientific and practical conference Modern information technologies and IT education, MSU, Moscow, 2015, vol 2, 409-414 (in Russian)

[11] Vostokin S.V., Doroshin A.V., Artamonov Y.S., Application of the Templet Web system to solve problems of mathematical modeling using high-performance systems. In Collected Works of the XVIII All-Russian Seminar on Motion Control and Navigation of Aircraft: Part II. Samara, (15-17 June 2015, Samara), Samara Science Center of RAS, 2016, 17-21 (in Russian)

[12] The Templet Markup Language on GitHub: https://github.com/Templet-language

[13] Artamonov Y.S., Using the EMMSP model to predict the available computing resources in the cluster systems. In Proceedings of the Samara Scientific Center of the Russian Academy of Sciences, Vol 18, 4(4), 2016, 681-687

[14] Artamonov Y.S., Prediction of Cluster System Load Using Artificial Neural Networks. In Proceedings of the International conference Information Technology and Nanotechnology. Session Data Science (Samara, Russia, 24-27 April, 2017). – CEUR Workshop Proceedings, 2017, 59–63

[15] Artamonov Y.S., Prediction of cluster system load using adaptive model mixture. International Journal of Open Information Technologies 5.5 (2017): 9-15

[16] Artamonov Y.S., Vostokin S.V., Development of distributed applications for data collection and analysis on the basis of a microservice architecture, Proceedings of the Samara Scientific Center of the Russian Academy of Sciences, Vol 18, 4(4), 2016, 688-693

[17] Litvinov V.G., Development and application of the computational model for skeleton solutions. Case study – using "bag-of-task" for HRBF neural network learning, Vestn. Samar. Gos. Tekhn. Univ., Ser. Fiz.-Mat. Nauki [J. Samara State Tech. Univ., Ser. Phys. Math. Sci.], 3(36), 2014, 183–195

[18] Stolbova A.A., Vostokin S.V., Popov S.N., Calculation of coefficients of wavelet transform on cluster systems. In: Advanced information technologies and scientific computing (PIT 2017): proceedings of the international scientific conference, ed. S.A. Prokhorov, Russia, Samara: Samara Scientific Center of RAS, 2017, 476-478 (in Russian)

[19] Sullivan W.T., Werthimer D., Bowyer S., Cobb J., Gedye D., and Anderson D., A new major SETI project based on Project Serendip data and 100,000 personal computers. International Astronomical Union Colloquium, vol 161, 1997, 729–734

[20] Mell P. M., Grance T., The NIST definition of cloud computing, 2011

[21] Cole M., Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming, Parallel Computing, vol 30(3), 2004, 389–406, DOI: 10.1016/j.parco.2003.12.002

[22] González-Vélez H., Leyton M., A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers. Software: Practice and Experience, vol 40(12), 2010, 1135–1160, DOI: 10.1002/spe.1026

[23] Lamport L., The temporal logic of actions. ACM Transactions on Programming Languages and Systems. 16 (3). May 1994, 872–923, DOI: 10.1145/177492.177726

[24] Ward M.P., Language-oriented programming. Software-Concepts and toolkits, vol. 15(4), 1994, 147–161.

[25] Kacsuk P., P-GRADE portal family for grid infrastructures. Concurrency Computat.: Pract. Exper., 23, 2011, 235–245

[26] Knyazkov K.V., Kovalchuk S.V., Tchurov T.N., Maryin S.V., and Boukhanovsky A.V., Clavire: e-science infrastructure for data-driven computing. Journal of Computational Science, vol. 3, no. 6, 2012, 504–510

[27] Sukhoroslov O., Volkov S., Afanasiev A., A Web-Based Platform for Publication and Distributed Execution of Computing Applications. In: Proceedings of 14th International Symposium on Parallel and Distributed Computing (ISPDC). IEEE, 2015, 175-184