Templet Web: The Experimental Use of Volunteer Computing Approach in Scientific Platform-as-a-Service Implementation

Sergei Vostokin, Yuriy Artamonov, and Danil Tsarev

Samara University, Samara, Russia
vostokin_sv@ssau.ru

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: (1) the implementation of "on-demand" access; (2) source code deployment management; (3) 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 the 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 the service development.

Keywords. Cloud Computing, Platform as a Service, Volunteer Computing, Scientific Computing, Automatic Parallel Programming, Cluster Computing

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. Moreover, rejecting high-performance parallel computing when sequential computing may be implemented can lead to incorrect results. For example, it is possible to get an incomplete study of a model's parametric space; to get an incorrect conclusion about the properties, adequacy and boundaries of the applicability of the model to the studied object.

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 often implement the models in the form of a traditional sequential program. We may presuppose the following three reasons for this limited use of high-performance computing.

Firstly, it is important to access high-performance resources "on demand" in low-budget projects, preliminary studies, or training. At the same time, the access to high-performance academic systems is usually associated with bureaucratic procedures and is not implemented remotely via the Internet. The offline access registration entails financial costs and time loss when renting computing power.

Secondly, the usual form of access to computing resources for the high-performance computing industry is terminal access over a secure channel based on SSH protocols. Along with great flexibility, which is an advantage for a system programmer, this form of access is inconvenient for a mathematician. It requires him to master the skills of system administration that are not typical professional skills for a numerical modeling expert. This leads to an increase in the expenditure of time and finances for the organization of calculations on the model.

Finally, an equally important problem is the development of a parallel program for a numerical algorithm. The traditional form of the representation of a numerical algorithm is a sequential program representation. This algorithmic representation is natural for the mathematician. However, at present, the theory of compilation gives no universal methods of transforming the algorithmic representation into binary code suitable for practical use in high-performance systems. Therefore, modern tools for the 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. The study of programming tools and equipment features involves time and financial costs.

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 factors considered above. This automation includes: (1) the automation of the access procedure for a high-performance system; (2) the automation of program deployment (code upload, building, starting, and control of the program execution, downloading the results); (3) automatic parallel programming for high-performance computing systems.

2 Research Methods

Development and maintenance of a service with the requirements stated above are associated with solving many technical and scientific problems. The core of our strategy of providing access to a remote computing system is the concept of volunteer computing [2]. The donors of computing resources in our system are academic cluster account holders. The system is implemented as PaaS (platform as a service) cloud service [3]. In 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 lead us to the need of solving the problem of forecasting the computational load of the cluster. A user who provides the access to a cluster should know the periods when the access can be granted without compromising his/her own projects and the overall cluster load. At the same time, a user who gets the access needs to know when the effective work on the cluster will be possible. In order to solve this problem, mathematical methods of forecasting and data mining are used [4, 5].

Automation of parallel programming in our service is based on the concept of algorithmic skeletons [6, 7]. This method implies the storage of a set of frameworks for controlling parallel computations. The frameworks can be extended with sequential code for a task. To specify the semantics of parallel execution for the algorithmic skeletons, we apply the version of the actor model [8] and define the model in the temporal logic of actions [9]. The syntax of skeletons is developed according to the language-oriented programming approach [10], considering the maximum compatibility with development environments and tools that are traditional for high-performance computing (C ++, OpenMP, MPI).

Below we consider the technologies developed and tested in the service, statistical data on its functioning, and numerical modeling tasks solved with the help of it. In conclusion, the results of service development are summarized.

3 On-demand Access to Academic Cluster

Our service uses the idea of volunteer computing to access a remote computer system. A Volunteer is a person who has an account on a remote system accessible via the SSH protocol. A Consumer is a person who does not have an account on the remote system, but wants to use the Volunteer's account. To run the Consumer's program through the Volunteer's account, our service implements the following protocol:

- Consumer submits the program source code and input data to Volunteer;
- Volunteer runs the program on his/her own behalf on the remote system;
- Volunteer returns result of the run to Consumer.

The system acts as a broker and assumes the following obligations:

- storing of source code, data and the result of computations for mutual audit of Consumers and Volunteers actions;
- organizing the access to this information by both Volunteer and Consumer;

- multiplexing access for many Consumers to the same 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 [11] volunteer computing middleware, in our implementation, multiplexing is used when connecting to Volunteer, not to Consumer. It is also assumed that the Volunteer gives access to a multitasking system (for instance, batch system).

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

4 Deployment Automation

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

A 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.

Task is performed in an Environment. Each Environment contains information for connecting to the Volunteer system. Environment implements the life cycle of Tasks in Volunteer system. Specific operations of the life cycle are defined in the Template. For example, 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 working in a project can be conducted using the browser or a version control system.

5 Automatic Parallel Programming

Let us demonstrate automatic parallel programming in our system using the example of a computation by the master-workers scheme. This scheme is typical for volunteer computations in BOINC. When you create Tasks from the master-workers 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, struct *task* is the input data of the task; struct *result* is the result of the calculation of the task; *proc* is the procedure for calculating the task performed by the workflow; struct *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 volunteer 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 [12].

The master-workers skeleton can be used to prototype the applications for the BOINC platform. Note that the integration of BOINC with cluster systems was studied earlier in the CluBORun project [13, 14]. In the future, it is possible to run master-workers applications from our system to the BOINC network, if we consider the BOINC control node as special type of Volunteer system.

Other skeletons, for example pipeline, are also implemented in our system. Our system has a DSL-based constructor for developing skeletons in the form of actor networks.

6 Results of Service Operation

The Templet Web service is deployed in the private cloud of the Supercomputer Center of Samara University [15]. The service is used for teaching students high-performance programming, for automatic parallel programming technologies research, and for solving applied problems using numerical algorithms [16].

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 on 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 allows you to control 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 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 editing code in a web browser was added to the system. As you can see from Tables 2-4, this function is in demand among users and 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 browser.

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

The system includes a universal skeleton constructor based on the Templet markup language [8]. The source code for the skeleton constructor and examples of its use are available online [17].

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 forecast of changing the cluster load, calculated by various mathematical methods: maximum likelihood method [18], neural networks [19], an adaptive combination of the listed methods [20]. The forecasting system is implemented using the microservice approach [21].

We use Templet Web system to solve problems in modeling the dynamics of space vehicles. The study of dynamics includes the numerical solution of the equations of spacecraft motion 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-workers calculations [22].

In this project, we also examined the applicability of the master-workers scheme to parallel algorithms implementation for training neural networks with the selection of the optimal structure of neurons in hidden layers [23]. A program for parallel continuous wavelet transform has been developed in the course of the study of the problem of analysis of acoustic signals from a cutting tool [24].

7 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 the automation of programming, on-demand access to resources and the automation of deployment. 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 by 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.

Acknowledgments. 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/ПЧ).

References

1. The Templet Project. http://templet.ssau.ru/
2. W.T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, and D. Anderson. A new major SETI project based on Project Serendip data and 100,000 personal computers. International Astronomical Union Colloquium, vol. 161, pp. 729–734, Jan. 1997.

3. P. M. Mell and T. Grance. The NIST definition of cloud computing, 2011.
4. J. Han, M. Kamber, and J. Pei, Data mining: concepts and techniques. Elsevier, 2011.
5. S. Makridakis, S.C. Wheelwright, and R.J. Hyndman. Forecasting: Methods and Applications, Third edition. John Wiley and Sons, 1998.
6. M. Cole. Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming, Parallel Computing, vol. 30, no. 3, pp. 389–406, Mar. 2004.
7. H. González-Vélez and M. Leyton. A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers. Software: Practice and Experience, vol. 40, no. 12, pp. 1135–1160, Nov. 2010.
8. S.V. Vostokin. Templet: a markup language for concurrent actor-oriented programming. CEUR Workshop Proceedings, 2016. vol. 1638. pp. 460-468.
9. L. Lamport. The temporal logic of actions. ACM Transactions on Programming Languages and Systems. 1994. vol. 16 (3). pp. 872–923.
10. M.P. Ward. Language-oriented programming. Software-Concepts and toolkits. 1994. vol. 15(4). pp.147–161.
11. D. P. Anderson. Boinc: A system for public-resource computing and storage. In Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on, pages 4–10. IEEE, 2004.
12. S.V. Vostokin, D.A. Tsarev. 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. pp 481-484.
13. A.P. Afanasiev, et al. Technology for integrating idle computing cluster resources into volunteer computing projects. Proc. of The 5th International Workshop on Computer Science and Engineering, Moscow, Russia. 2015.
14. O. Zaikin, M. Manzyuk, S. Kochemazov, I. Bychkov, and A. Semenov. A volunteer-computing-based grid architecture incorporating idle resources of computational clusters. In International Conference on Numerical Analysis and Its Applications, pp. 769-776. Springer, Cham, 2016.
15. Supercomputing Center of Samara University. Templet Web. http://hpc.ssau.ru/node/3130
16. Y. S. Artamonov, S.V. Vostokin 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. - pp. 409-414.
17. The Templet Markup Language: a tool for concurrent, actor-oriented, skeleton programming. https://github.com/templet-language
18. Y.S. Artamonov. Using the EMMSP model to predict the available computing resources in the cluster systems. Proceedings of the Samara Scientific Center of the Russian Academy of Sciences, vol. 18, № 4(4), 2016. pp.681-687.
19. Y.S. Artamonov. Prediction of cluster system load using artificial neural networks. In Proc. of ITNT-2017, Samara, 2017.
20. Y.S. Artamonov. Prediction of cluster system load using adaptive model mixture. International Journal of Open Information Technologies 5.5 (2017): 9-15.
21. Y.S. Artamonov, S.V. Vostokin. 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. pp.688-693.
22. S.V. Vostokin, A.V. Doroshin, Y.S. Artamonov. 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. pp.17-21.

23. V.G. Litvinov. 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.], 2014, no. 3 (36), pp. 183–195.
24. A.A. Stolbova, S.V. Vostokin, S.N. Popov. Calculation of coefficients of wavelet transform on cluster systems. 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. pp 476-478.