Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 136 (2018) 44–51

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Young Scientist Conference on Computational Science

LiFlow web service for quick launch of large experiment series on supercomputers

Evgeniy Kuklin[a,b], Sergei Pravdin[a,b]

[a] *Krasovskii Institute of Mathematics and Mechanics, Ekaterinburg, Russia*
[b] *Ural Federal University, Ekaterinburg, Russia*

Abstract

In many fields of research, for example in simulations of living systems, the automatic execution of many numerical experiments with varying parameters is a crucial task. We propose a web-based multiplatform service for launching a series of experiments on a supercomputer. The service interface allows the user to set parameters and run jobs on a supercomputer without working in the command line. The system minimizes the number of user actions required for launching a large series of experiments. Using the built-in database, one of the previously conducted experiments can be picked up and quickly restarted with new settings. We provide the architecture of interaction between a computing cluster, a database, and an external web application, as well as the algorithm for generating tuples from sets of values of individual parameters, and features of the graphical user interface. The system actively carries out computational experiments to study the drift of spiral waves in the myocardium on the *Uran* supercomputer, Ekaterinburg.

Keywords: parallel computing systems; supercomputers; experiment series execution; computational experiment reproducibility; GUI; living system simulation

1. Introduction

Various fields of research, especially the simulation of living systems, often require numerous computational experiments on the same model with different parameters and values, requiring significant computational resources. This difficulty leads to four primary issues. Firstly, since experiments are computationally exhausting and time-consuming, they are difficult to conduct in a reasonable amount of time without parallel computing systems and supercomputers. Secondly, hundreds of numerical experiments must be prepared and performed. Thirdly, the experiments have to be reproducible by others. Lastly, many scientists are not ready to work solely with command line on supercomputers.

E-mail address: key@imm.uran.ru

10.1016/j.procs.2018.08.236

Quite often, researchers use existing specialized application packages and access them from various web platforms created around the world (the SaaS model). However, most of the packages are designed for a specific task and use pre-installed software. In addition, they do not allow users to flexibly configure the automatic generation of input data for different simulation models.

To partially solve these problems, we offer the web-based service for easy launching a series of experiments on a supercomputer. The distinctive feature of the system is aiming at automatically conducting a large series of computational experiments on the same model for different parameter values, saving scientists from manually preparing the input data and launching jobs. Also, it tracks and saves parameters of previous experiments to be quickly restarted as-is, or with new settings. The service provides a simple web interface to work from any computer without installing additional software.

The project was created for examining models of hearts, so we called it LiFlow (LIving system simulation workFLOW). However, it is also applicable to any computational cluster software using a standard *ini* configuration file as input.

2. Related work

Currently, high-performance computing and storage systems solve most of the computational scientific problems independently, pushing out grid system into the past. Therefore, when developing middleware, the focus has shifted to building convenient and efficient means for accessing individual computing resources.

The principal development path of modern information technology is aimed at facilitating access to resources for the end user and reducing the time and financial costs of using web technologies. In many cases, researchers need to run a large number of similar computing tasks to solve a problem in a particular application area. Quite often, these tasks are performed by using existing specialized application packages, the tasks being identical in form and differing only in the values of the parameters. In those cases, the pre-installation and configuration of commonly used application packages and resources, the availability of specialized web services, and appropriate user-web interfaces (the SaaS model) allow the owners of computing resources to increase efficiency [1].

A number of projects are used to integrate application software packages with supercomputers. For example, the Distributed Virtual Test Bed (DiVTB) platform [2] provides a task-oriented approach for solving specific classes of problems in computer-aided engineering, through resources supplied by grid computing environments. It has a user-friendly graphical user interface where the parameters of a computational experiment can be specified, and the experiment can then be executed on a supercomputer.

Another platform, the Education-research Integration through Simulation On the Net (EDISON) [3] is designed and implemented to access and run various technology computer-aided design software tools. The platform provides an easy-to-use GUI that helps geographically-distributed researchers to run and share their tools in five areas: computational fluid dynamics, computational chemistry, nano physics, computer-aided optimal design, and computational structural dynamics.

Sometimes, researchers and engineers need not only to conduct an experiment but also to upload and share datasets and documents, prepare reports and articles, ask questions and post responses, write the source code working together in a private space. The combination of all these activities into a single system is called a web laboratory. The perfect example of such a laboratory is UniHUB [4], which is an extension of the OpenStack platform.

The multifunctional platform CLAVIRE is designed for the efficient management of computing, information, and software resources of distributed heterogeneous computer infrastructures within the cloud computing model [5]. It can be used to provide high-performance domain-specific services within the cloud computing model with the possibility of utilizing existing distributed computing infrastructure resources. The CLAVIRE basic application software toolkit includes dozens of packages in different areas.

Often the output of a package is the basis for the generation of the task for the next application package. Such series of tasks are called workflows. Thus, various scientific computation workflow systems like Pegasus [6] are developed. Nevertheless, we are dealing not with a complex workflow, but with multiple repetitions of one step.

A distinctive feature of LiFlow is providing users with the ability to use any software to run experiments, while many systems are limited only by an integrated set of algorithms or applications or bounded to particular computing resources. Also, most of the projects do not provide automation, such as launching of a series of computational

experiments with varying parameters. However, it is worth noting the positive experience of adapting such systems to the management via the web interface, which is what inspired us to create our own.

3. Logical architecture

The architecture of the computational experiment can be represented by four layers: user, software, computational, and storage. In order to ensure the effective usage of modeling applications, an intermediate layer between the software and the user is sometimes introduced. Adding an intermediate layer means converting general application software into "software as a service." A service layer allows us to flexibly configure the system by setting, processing, and generating input data, as well as providing parameters to save for reuse and support for reproducibility. The LiFlow service layer consists of the web interface, the database, and the special algorithm for generation of configuration files.

3.1. Web interface

After facing technical problems in a previous version of the project [7] when using Python for GUI development, we decided to change the concept of the interface. The use of different operating systems and the possible lack of administrator rights lead to problems with installing the required packages for the desktop version. Therefore, we decided to use a more generic web-based interface, which is highly likely to work on any browser or computer, even on a tablet or a smartphone.

The process of launching a series of experiments in the LiFlow system is divided into four steps. In the first step (Fig. 1), the user can start a new project, or select one of the previous projects, as well as check the status of jobs already running on the supercomputer.

The second step (Fig. 2) involves editing the configuration file for the simulation program. The key aspect of modeling is the launch of numerous experiments on one model while altering several variables. For user convenience, the values for the varied parameters can be specified in one file by separating the values with spaces.

In the third step (Fig. 3), the user specifies all the necessary information to place the jobs in the execution queue on a parallel computing system, such as the path to the source folder, the parameters for the workload manager, the working directory, and the brief description of the experiment. It is assumed that the source code is already compiled on the supercomputer.

The last step displays the details of the series of experiments, as well as the queue of the active user jobs to confirm that the tasks have successfully started for calculation. To display the queue, the supercomputer workload manager receives a request from the web server.

3.2. Service database

The reproducibility of computational experiments is a very important part of research in the field of modeling. Providing reproducibility is a complex process that requires capturing of all input data of a particular experiment, the used libraries, and the configuration of the hardware. The LiFlow system saves all the necessary input data and parameters, which ensures reproducibility if the hardware has not significantly changed.

Every experiment is assigned a unique identifier at the preparation stage. It is stored in the database along with the other parameters of the experiment and becomes the name of the directory storing the experiment-related files. These files are created and saved as the steps required to run the experiment are performed.

On the service home page, the latest experiments of all users of the system (using information from the database) are displayed (see Fig. 1). This makes it possible to avoid duplicating of calculations, when the required experiment has already been conducted by someone, or to reproduce the results of one's colleagues.

Moreover, once performed, a series of experiments can be easily restarted with updated parameters after the system has filled all other fields, which accelerates repeated runs of calculations. We use the PostgreSQL [8] database for the project. The scripts in the LiFlow system are written in Python, Bash, and PHP.

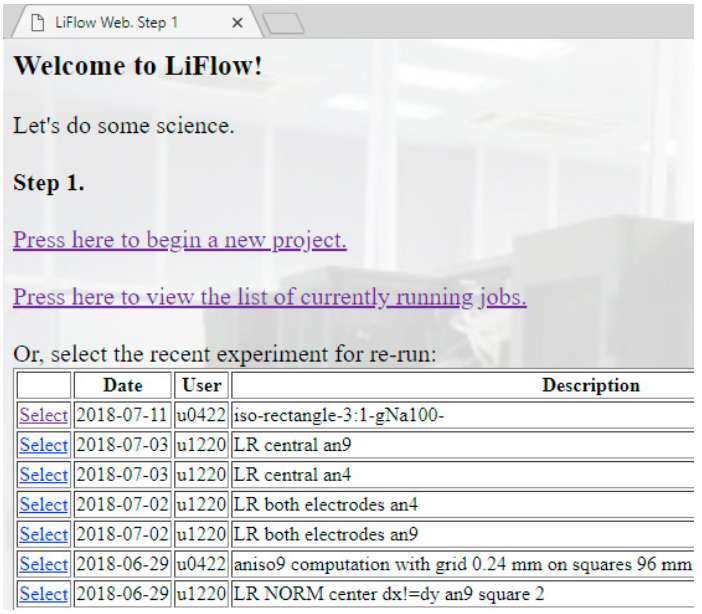

Figure 1: LiFlow system GUI. Step 1.

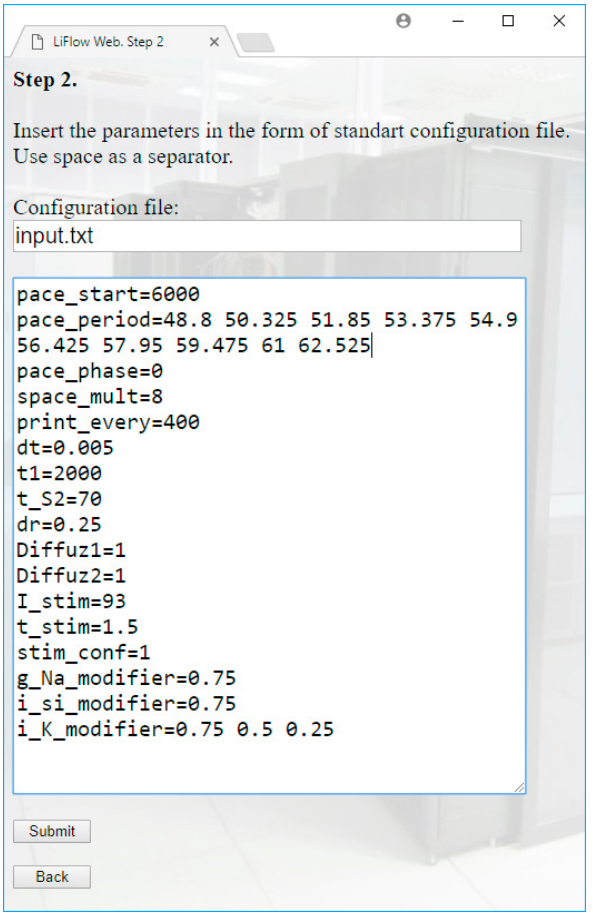

Figure 2: LiFlow system GUI. Step 2.

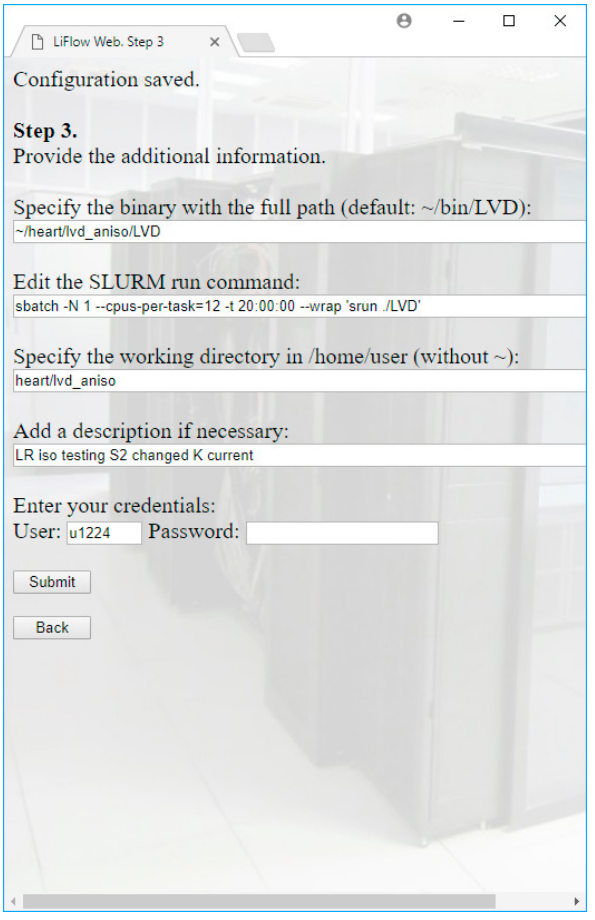

Figure 3: LiFlow system GUI. Step 3.

3.3. Generation of configuration files

The simulation software stores all model parameters in a standard *ini* configuration file. For example, to study the drift of spiral waves in the myocardium, several models with 10–20 parameters are used, and almost every parameter in the corresponding configuration files can be variated. The system never knows how many combinations will result from user settings: a recursion is not suitable here. So, an efficient algorithm for full parameter enumeration was necessary.

Each input file describes values of parameters $a_0, a_1, \ldots, a_k$. According to a specific task, a parameter a_i can be constant for all launches or must take values from a set $V_i = \{a_i^0, a_i^1, \ldots, a_i^{n_i}\}$. So, a set of input files where varying parameters take values from their respective sets must be created. We propose a simple algorithm to get all tuples.

The number of tuples is $N = (n_0 + 1)(n_1 + 1)\ldots(n_k + 1)$. Each input file is created based on indices $i_0, i_1, \ldots, i_k$ of parameter values. To get a tuple corresponding to an index $I \in \{0, \ldots, N-1\}$, we use a backing array $J_{1\ldots k}$ and utilize the following algorithm.

Algorithm

procedure FindTuples
$J_k = I$
for p from $k-1$ downto 1 do
begin
$i_p = J_{p+1} \bmod n_p$
$J_p = J_{p+1} \operatorname{div} n_p$
end
$i_0 = J_1 \bmod n_0$

Finally, the I^{th} tuple is formed using indices $i_0, i_1, \ldots, i_k$, which serve to create N configuration files with unique parameter values. Each configuration file is placed in its own directory, from where its copy of the simulation software is launched. For convenience, the directory names depend on the values and names of the varied parameters.

4. Service technical structure

One of the primary objectives of the project is making an easy-to-use service with a user-friendly interface that requires minimal adaptation. As mentioned above, it was decided to use the universal web interface, which will work on any browser of almost any computer.

The schematic diagram of the system is shown in Fig. 4. The main components of the system are located on a dedicated server. These components are: a web server, which provides the interface, a database, a storage directory for experimental metadata, and several service scripts.

The service uses the concept of a computational package, which is prepared via the interface and then sent through SFTP protocol to the supercomputer storage system into the user's home directory. The entire computational package is stored in the special directory on the web server for possible reuse in the future; the key information about it is written to the database. All supporting scripts are sent along with the original data in the computational package, but they can also be taken from the storage system.

On the supercomputer, after the automatic preparation of input configuration files for each experiment of the series, the launch script interacts with the supercomputer workload manager, such as SLURM [9], to set the tasks for execution.

Outgoing data remains in the storage system, which is connected with the computational nodes and the cluster head node via NFSv4 protocol. These data also can also be placed in the common archive of a research group.

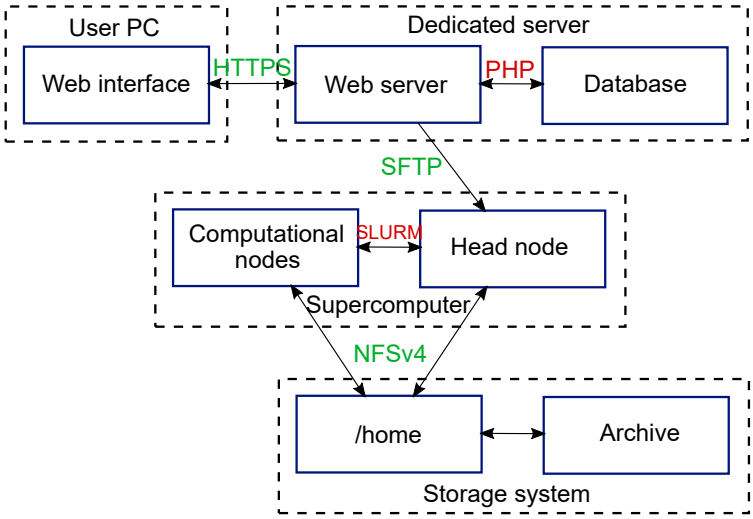

Figure 4: LiFlow web service structure.

5. Experiments

The users appreciated the convenience of the new LiFlow web GUI and the ability to launch a series of dozens of experiments in five minutes without working in the command line. In the case of a large series of experiments, without using the system, the manual preparation of data could take up to half an hour. In this case it is easy to get confused with parameters. Using the saved settings of previous experiments, the time to launch a new experiment series can be reduced to less than a minute. Overall, our service helped the researchers to conduct computational experiments more efficiently.

The proposed system was tested with the *Uran* supercomputer located at the Krasovskii Institute of Mathematics and Mechanics of the Russian Academy of Sciences in Ekaterinburg. The service is actively used for carrying out computational experiments to study the drift of spiral waves in the myocardium.

Myocardium is an active medium, which means that it consists of interconnected elements; they have one resting state and can temporarily come to an excited state when a stimulus of enough power is applied. Excited elements produce stimuli that spread in all directions and can excite neighboring elements, causing waves of excitation to appear. The waves can be plane and spiral. The normal work of a heart is caused by the plane waves.

The spiral waves in the myocardium emerge only in the case of dangerous arrhythmias and have to be treated if they persist. The usual method of treatment is to deliver a large current by a defibrillator, which completely resets excitation of the heart. Another method (called low-voltage cardioversion-defibrillation – LVCD) is the periodic applying a small electrical current to an area in the myocardium so that the stimulated zone produces plane waves with a greater frequency than the spiral wave. The plane waves begin to control a broader and broader zone, until they finally supersede the spiral wave. The aim of the study is to find out the optimal stimulation parameters for the LVCD. Five models with different numbers of parameters are used for the simulation.

An example of such modeling is the study of the wave drift in an anisotropic tissue using two cell models [10]. The myocardium has fibers, along which the excitation spreads faster than across them, so the fiber direction is an additional varying parameter (Fig. 5). Totally, $7 \cdot 5 \cdot 2$ parameter sets were examined and differences with the isotropic case were indicated. As a result, effective periods of stimulation and the positions of electrodes were found.

Another example is the work that was devoted to studying LVCD in anisotropic myocardium models with curved fibers using biophysical ionic Luo-Rudy cell model (Fig. 6) [11]. In total, 17 parameter sets were investigated. After measuring the time of the spiral wave drift and determining the type of interaction between waves, the findings were compared with the results of the isotropic and parallel-fibers anisotropic cases. Results of those simulations were similar. The dependence between the stimulation period and the drift velocity components was studied. However, new

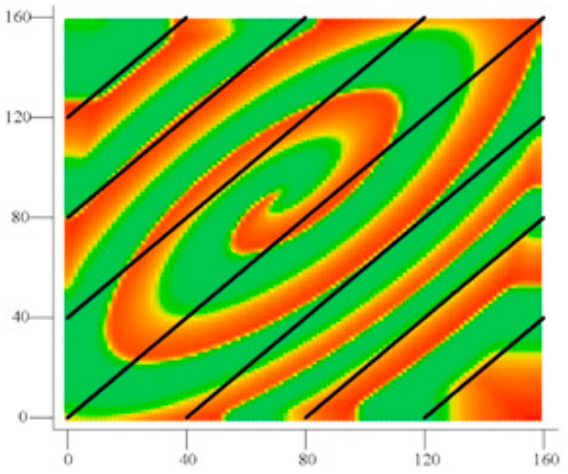

Figure 5: Diagonal fiber direction in the anisotropic tissue [10].

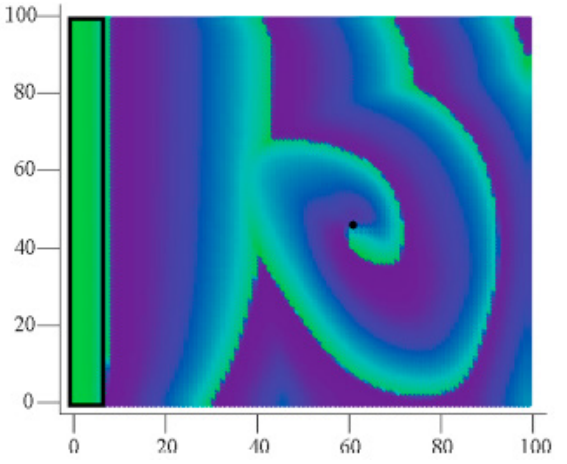

Figure 6: Interaction between the wave from the line electrode and the spiral wave in the anisotropic tissue with curved fibers [11].

spiral waves emerged, in many cases depending on the fiber direction and electrode configuration. Therefore, research on media with curved fibers is a task for future work.

All the aforementioned series of experiments were carried out using the LiFlow web service. The total amount of conducted series stored in the database is over 200.

Though the service has only been used for the heart modeling problems, it can be applied to run on the computational clusters of any software, especially those written by researchers themselves. The service is also suitable for other classes of tasks in which the use of a supercomputer provides significant advantages. For example, it can be used to construct the receiver operating characteristic curve utilized to estimate the quality of the binary classification in pattern recognition and machine learning. In this case, the service allows one to simultaneously run an algorithm accuracy estimation with different thresholds using all available supercomputer nodes without programming. Also, the service can be used to select optimal parameters for neural networks and other algorithms.

6. Conclusion and future work

The paper presents the web-based multiplatform service LiFlow for launching large series of experiments on supercomputers. The service interface allows the user to set the parameters and run the jobs on a supercomputer without working in the command line. The system focuses on minimizing the number of user actions required for launching a large series of experiments. Using the built-in database, one of the previously conducted experiments can be picked up and quickly restarted with new settings.

The architecture for the interaction between a computing cluster, a database, and an external web application, as well as the algorithm for generating tuples from sets of values of individual parameters, and features of the graphical user interface is provided. The system is actively used for carrying out computational experiments to study the drift of spiral waves in the myocardium on the *Uran* supercomputer.

Future plans include the option of automatically writing to the archive while indexing to the database as well as the ability to search by keywords through previously launched experiments. Adapting the system for more interactive job control would greatly enhance usability. There are also some associated jobs, like moving to a more convenient framework.

Acknowledgements

The work is supported by the RSF project 17-71-20024 (IMM UB RAS). The study was performed using the *Uran* supercomputer of the Krasovskii Institute of Mathematics and Mechanics.

References

[1] Stanislav Polyakov, Andrey Demichev, and Alexander Kryukov. Web toolkit for scientific research: State of the art and the prospect for development. *Procedia Computer Science*, 66:429–438, 2015.

[2] Gleb Radchenko, Elena Hudyakova, and Evgeniy Zakharov. Providing a web portal for development and utilization of distributed virtual test beds on the basis of unicore grid infrastructure. *UNICORE Summit 2013: Proceedings, 18th June 2013, Leipzig, Germany*, 21:57, 2013.

[3] Suh Young-Kyoon, Hoon Ryu, Hanki Kim, and Kum Won Cho. EDISON: A web-based hpc simulation execution framework for large-scale scientific computing software. In *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*, pages 608 – 612. IEEE, 2016.

[4] O Samovarov and S Gaysaryan. The web-laboratory architecture based on the cloud and the UniHUB implementation as an extension of the OPENSTACK platform. *Trudy Instituta sistemnogo programmirovaniya RAN*, 26(1), 2014.

[5] Konstantin Knyazkov, Sergey Kovalchuk, Timofey Tchurov, Sergey Maryin, and Alexander Boukhanovsky. CLAVIRE: e-science infrastructure for data-driven computing. *Journal of Computational Science*, 3(6):504–510, 2012.

[6] Ewa Deelman, Karan Vahi, Mats Rynge, Gideon Juve, Rajiv Mayani, and Rafael Ferreira da Silva. Pegasus in the cloud: Science automation through workflow technologies. *IEEE Internet Computing*, 20(1):70–76, 2016.

[7] Evgeniy Kuklin, Andrey Sozykin, Konstantin Ushenin, and Dmitriy Byordov. LiFlow: A workflow automation system for reproducible simulation studies. *CEUR Workshop Proceedings*, 1839:208 – 217, 2017. URL `http://ceur-ws.org/Vol-1839/MIT2016-p19.pdf`. Proceedings of the International Conference Mathematical and Information Technologies, Vrnjacka Banja, Serbia - Budva, Montenegro, August 28 - September 5, 2016.

[8] PostgreSQL: The world's most advanced open source database. `https://www.postgresql.org/`. Accessed: 2018-05-05.

[9] M.A. Jette, A.B. Yoo, and M. Grondona. SLURM: Simple linux utility for resource management. *Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP)*, 2862:44 – 60, 2003.

[10] Timofei Epanchintsev, Sergei Pravdin, Andrey Sozykin, and Alexander Panfilov. Simulation of overdrive pacing in 2D phenomenological models of anisotropic myocardium. *Procedia Computer Science*, 119:245 – 254, 2017. ISSN 1877-0509. URL `http://www.sciencedirect.com/science/article/pii/S187705091732392X`. 6th International Young Scientist Conference on Computational Science, YSC 2017, 01-03 November 2017, Kotka, Finland.

[11] Timofei Epanchintsev, Sergei Pravdin, and Alexander Panfilov. Spiral wave drift induced by high-frequency forcing. Parallel simulation in the Luo–Rudy anisotropic model of cardiac tissue. *Lecture Notes in Computer Science*, 10860:378–391, 2018. International Conference on Computational Science, ICCS 2018, 11-13 June 2018, Wuxi, China.