LiFlow: A Workflow Automation System for Reproducible Simulation Studies

Evgheniy Kuklin[1,2,3], Andrey Sozykin[1,2,3], Konstantin Ushenin[2,3], and Dmitriy Byordov[2,3]

[1] Krasovskii Institute of Mathematics and Mechanics, Ekaterinburg, Russia
[2] Institute of Immunology and Physiology UrB RAS, Ekaterinburg, Russia
[3] Ural Federal University, Ekaterinburg, Russia
key@imm.uran.ru
http://www.ipgg.sbras.ru

Abstract. Simulation of living systems often requires numerous computational experiments on the same model for different parameter values. This paper describes the design of a user-friendly workflow automation system LiFlow for simulation of living systems, which is capable of conducting such a large series of computational experiments on supercomputers. The system provides a convenient interface for preparing input experimental data, executing the experiments on a supercomputer, and storing experimental results in a storage system. A distinctive feature of LiFlow is its simplicity and usability—the system is intended to be used by researchers in mathematical biology and biophysics without extensive knowledge in parallel computing. The paper provides examples of the use of the LiFlow system for simulation of the human heart left ventricle.

Keywords: parallel computing systems, supercomputers, living system simulation, computational workflow, computational experiment reproducibility.

1 Introduction

Simulation of living systems requires significant computational resources. Such investigations are certain to be rather time-consuming and, thus, are hard to be conducted in a reasonable time without parallel computing systems and supercomputers. However, the use of parallel computing systems requires a high degree of qualification in computer science, which many researchers involved in living systems modeling do not possess or want. Moreover, the data preparation for computational experiments is routine and time-consuming. The user needs to copy data to a supercomputer, compile the source code if necessary, enqueue the jobs with the supercomputer resource manager, and keep track of their completion. Such routine tasks should be automated.

Living system simulation often demands a large number of computational experiments on the same model but with varying parameter values. Nowadays, researchers have to prepare the configuration, input data, and the desired parameter values, and then separately execute the simulation software for each

experiment. With the number of required computational experiments increasing up to hundreds or even thousands, which is typical for living systems simulation, manual preparation and execution of experiments becomes very labor-intensive and often nearly impossible. In such a case, scientists often execute only a fraction of the required computational experiments, which negatively affects the research results, hence the need to automate the execution of series of computational experiments with varying parameter values.

Another problem that researchers in simulation of living system often face is non-reproducibility of computational experiments. This problem is directly related to the large number of computational experiments that scientists have to carry out in order to obtain meaningful results. Due to pressure for publishing, scientists devote little time to keeping the records of experimental details, especially in case of hundreds or thousands of experiments. In addition, many other factors can affect computational results, such as a change in the version of the compiler or a required library on the supercomputer. Automated recording of experimental details and storage of simulation results can help to ensure reproducibility of the computational experiments.

We developed LiFlow (LIving system simulation workFLOW), a workflow system that addresses this need for automation. LiFlow provides the scientists with a convenient graphical user interface (GIU) that allows to prepare and execute a series of computational experiments on a parallel computing system with a single click.

One of the important goals of creating the LiFlow system was to make the initial learning process of the workflow tool very simple. Otherwise, busy scientists will not invest their time in studying the capabilities of the new system, and it will be useless.

The LiFlow system is primarily intended for simulation of living systems; we provide some examples of using LiFlow to simulate the human heart left ventricle. However, LiFlow could also be used in other areas that require conducting a large number of computational experiments on parallel computing systems.

2 Related Work

To bridge the gap between researchers and software engineers and reduce experiment preparation time, scientific computation workflow systems [1] are being developed. The most frequently used among them are Taverna, Kepler, and Triana. Taverna [2] is an open source workflow system particularly focused on bioinformatic applications and services; it is based on the XScufl language. Kepler [3] is a scientific workflow system that builds on the PtolemyII system, which is a visual modeling tool written in Java. Triana [4] is a GUI-based workflow system for coordinating and executing a collection of services. All these tools have some visual interfaces that allow graphical composition of operations. The systems provide the ability to integrate distributed computing resources, applications, data sets, and tools for computational experiments. In addition, the systems hide the complexity of distributed computing systems from users,

enabling them to describe the workflow graphically. The existing systems for scientific computing workflows are able to use the computing resources of various types (GRID, supercomputers, distributed systems, etc.), data stores (local, network, cloud), and tools (visualization, statistical processing, etc.); they also include provenance tracking, either as an integral part or as an optional module. As a result, such systems are very complicated and difficult to install, maintain, and use. Their main disadvantage is the fact that creating a new component can require considerable efforts and a detailed knowledge of the workflow system architecture. However, when simplified workflows are sufficient, there is no need for unwieldy options with a lot of settings. On the contrary, computational experiments should preferably launch "in one click."

Another possible solution to the problem is to use an environment that provides the integration of application software packages with supercomputers. An example of such system is DiVTB [5], which provides user-friendly graphical interface where parameters of a computational experiment can be specified, and the experiment can then be executed on a supercomputer. However, such systems do not provide automation of the tasks that are popular in living systems modeling—such as launching of a series of computational experiments with the same model but varying parameter values; also, they do not support metadata tracking.

To solve the reproducibility problem special software tools can be used. They provide the ability to automatically capture and store for future use all the environment of a computational experiment, such as the simulation software, the input and output data, the hardware and software configuration of the computing system, etc.

There are two basic methods used by the reproducibility improvement tools. One method is based on executing experiments in a virtual environment, such as virtual machines or cloud [6]. After an experiment completes, the snapshot of the virtual machine is saved together with the simulation software, the output data, the experimental log, and so on. Furthermore, the snapshot can be made publicly available; other scientists can use it to reproduce the experiment and cite in their papers. Unfortunately, this approach is not suitable for parallel computing systems because virtualization considerably reduces the performance of such systems. In addition, such approach will require capturing the snapshots of all nodes in the cluster that were used for running the experiment, which is not feasible.

The second method is based on capturing the snapshot not of the entire virtual machine but of the simulation software executable and the output data. This approach is used in the CDE system (Code, Data, and Environment packaging) [7]. However, a package prepared by the CDE system depends on the software configuration of the computational system. Although the configuration of a personal computer or a virtual machine is relatively easy to replicate, it can be very difficult to adjust the configuration of a parallel computing system. Most of such systems are shared among a great number of users; only qualified

administrators can install or configure the software. Hence, such approach is also not suitable for parallel computing systems.

In order to ensure the reproducibility of computational experiments, we aimed at integrating LiFlow with Sumatra [8], which is an open source tool to support reproducible computational research. The Sumatra system [8] aims to capture the information required to recreate the computational experiment environment instead of capturing the experimental context itself. Sumatra uses the source code of the program instead of the binaries, stores the logs of the compilation process, and saves the information about all dependencies and general operating system configuration. Furthermore, Sumatra provides the ability to store the output data for future use in a database. In addition, Sumatra allows to index and search the data about experiments carried out, including additional information provided by scientists. For example, if experimental data was published, scientists can add tags with the name of the paper (and, perhaps, additional information such as the figure or table with the data) to the experiment record in the catalog. This allows researchers to quickly find the information required to reproduce the experiment they are interested in among a large number of experiment records. Unfortunately, Sumatra lacks a convenient desktop user interface. Although Sumatra is a standalone project, it can be used as a library for third-party development and has its own API. LiFlow can use Sumatra for capturing and storing the information of previously conducted experiments in the database.

3 LiFlow system

3.1 Workflow

Workflow in the LiFlow system corresponds the one shown in Fig. 1. During the first stage, researchers prepare the description of the so-called experiment series, which is a set of experiments with the same model and varying parameter values. The preparation includes the selection of simulation software that will implement the required model, generation of the configuration files with the required parameters, and creation of the input data files for each experiment. Next, the experiments are launched on a parallel computing system.

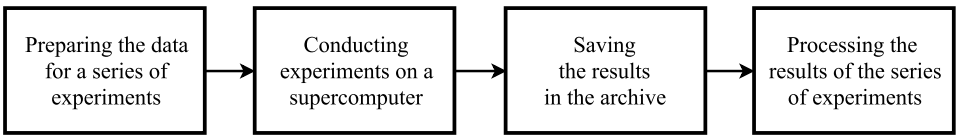

Fig. 1. LiFlow system workflow

When the experiments are completed, the obtained results are automatically stored in the archive in a form ready for processing (visualization, statistical

processing, etc.). Thus, a user is only required to create a description of the experiment series, all the rest is done automatically. In addition, the user is able to process the results of experiments from the archive manually using third-party tools.

3.2 Computational Package

Similarly to the CDE system, LiFlow uses the concept of a computational package that contains all the information required to execute a series of experiments. The LiFLow computational package consists of the following components:

- Source code of the simulation software, which can be loaded from a code repository of a version control system, such as Git.
- Generator of experiment series that describes how to generate the desired parameter values for the experiment series.
- Initial data and parameters to launch the simulation software.

A distinctive feature of the LiFlow computational package is that it describes not one experiment but a whole series of experiments. Each experiment in the series uses the same simulation software but different values of the model parameters. The parameter values for every experiment are produced by the generators, which are a part of the software package that is based on the rules specified by the user.

3.3 LiFlow Architecture

The LiFlow system consists of the four main components (Fig. 2). The Computational Package Preparation Tool and the Experiment Execution GUI are installed on the researcher's personal computer, while the Experiment Execution Engine and the Parallel Computing System Adapter are deployed to the parallel computing system.

A user creates a computational package with the help of the Computational Package Preparation Tool and uses the Experiment Execution GUI to transmit the package to the desired parallel computing system and run the experiment series. Experiment Execution Engine on the computational cluster receives the package, compiles the source code of the simulation software, and executes the generator of the experiment series to produce a set of input data files for simulation software with various parameter values. Next, the set of computational jobs is generated with the same simulation software but different input files. The jobs are queued on the computational cluster using the Parallel Computing System Adapter, which interacts with the resource manager of the cluster.

Once the job is completed, the results of the experiment are automatically recorded to the Experiment Archive on the storage system. After all the jobs in the experiment series are completed, the Experiment Execution Engine sends an email with the report on the experiments' execution to the user.

The planned Sumatra module would be able to capture the environment of the computational experiment and store it in the Experiment Catalog in order

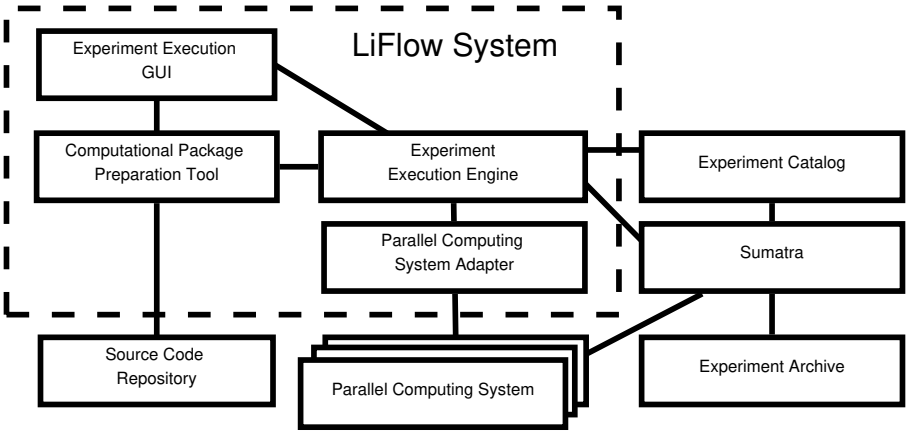

Fig. 2. LiFlow system architecture

to share the initial data and simulation results among the researchers and to improve the experiments' reproducibility.

4 Technical Details

The first stage of the LiFlow system implementation has been currently completed. The computational package in the implementation is represented by a directory in a file system that contains the subdirectories with the following components: the source code of the simulation software, the generator of the experiment series, the initial data for the generator, and the script for executing the experiments.

In the current implementation, the generator is a script that creates a series of experiments by varying the parameters in the configuration file of the simulation software. The LiFlow system supports two options for specifying the parameter values:

- The range of the parameter: an initial value, a final value, and an increment. One record in a configuration file of the generator produces the input data for several experiments.
- The explicit parameter values declaration. The parameter values must be specified for each experiment in the series.

The prepared computational package is transferred to the parallel computing system using the SSH or SFTP protocols. Next, the source code of the simulation software is built on the computational cluster. If the build process fails, LiFlow warns the user and sends back to him the build log file. In the case of a successful compilation, the system runs the generator of the experiment series to produce the input data for the experiments.

Currently, only one version of the Parallel Computing System Adapter is implemented, which is based on the SLURM Workload Manager [9]. The experiment startup script from the computational package enqueues the generated tasks for the series of experiments in the SLURM job queue. After the job is complete, the LiFlow system copies the output data to the Experiment Archive using the NFS protocol.

The scripts in the LiFlow system are written in Python. The storage of simulation software source codes is implemented as a Git repository provided by a third-party service.

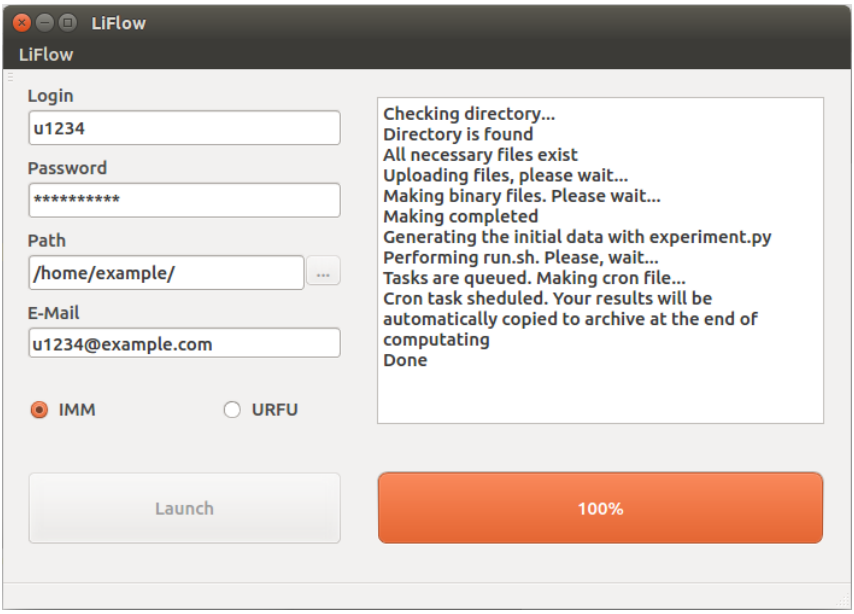

Fig. 3. LiFlow system GUI

Users are provided with a simple graphical interface, which allows one to execute a series of experiments on a parallel computing system in one click (Fig. 3). The user needs to select the parallel computing system to perform the computation, specify the credentials (login and password), the path to the folder with the computational package, and the email address (for job completion notifications). When the user clicks the *Launch* button, the LiFlow system starts the workflow process. The text output shows the current stage of the process of setting up the experiment and, if an error occurs, specifies where did it happen. Fig. 3 demonstrates an example of a successfully submitted experiment. The LiFlow GUI is also written in Python using the PyQt4 library and is designed to work both on Windows and Linux.

A disadvantage of the current LiFlow implementation is the lack of a failover mechanism. If an error occurs, the experiment will not be repeated. This approach is chosen because the failure can be caused not only by problems with hardware or system software, but also, more frequently, by an error in the simulation software or a wrong combination of parameters, for which the computation cannot be performed. In such a case, restarting the experiment will not lead to solving the problem, it will only unnecessarily load the computational cluster. Still, a failure in carrying out one experiment does not lead to termination of the entire experiment series.

5 Using LiFlow for Heart Simulation

Nowadays, the LiFLow system is integrated with the *URAN* supercomputer at the Krasovskii Institute of Mathematics and Mechanics and the computational cluster of the Ural Federal University. The system had been used on these clusters to execute several experiments in simulation of human heart left ventricle (LV) using the LeVen simulation system [10].

The study of the influence of the fiber direction in the LV anatomical model on the speed and consistency of its electrophysiological activation was performed using the LiFLow system [11]. A series of 55 experiments was performed, where two parameters, corresponding to the direction of the fiber course in electrophysiological models, were varied.

The same system can be used to reproduce the results of the research manually conducted before. Two series of experiments were performed in the investigation of the excitation speed of the LV myocardial tissue by using an anatomical model that allows to change the shape of the ventricle and the direction of the fiber course in it [12]. In one of the series of experiments, the area of the initial activation, the fiber direction of the anatomical model, and the ratio of coefficients in the diffusion tensor of the electrophysiological model were varied. In total, the work was based on more than 36 experiments with the parameter values generated by certain rules.

The paper [13] describes the research in the dynamics of the spiral waves in the LV of the human heart model with different geometry and direction of the fiber course. In the research, several series of experiments with the anatomy that approximate normal and pathological anatomy of the LV were carried out. In each series of experiments, the following parameters were varied: the thickness of the top, the value of the diffusion tensor, and the place of the initial start-up wave. In total, the work was based on more than 84 experiments.

6 Discussion

The users of the LiFlow system, researchers in mathematical biology and biophysics from the Institute of Immunology and Physiology UrB RAS, provided a generally positive feedback. Before, they needed approximately 30 minutes to

manually prepare and run one computational experiment on a parallel computing system. With the help of the LiFLow system, they could execute a series of dozens or hundreds of experiments in less than one hour. The users appreciated the convenience of the LiFlow GUI and the ability to obtain the results of simulation from the storage system. As a result, they do not have to deal with the Linux operating system on the computational cluster, which is unfamiliar to them. Overall, LiFlow helped the researchers from the Institute of Immunology and Physiology UrB RAS to conduct computational experiments more efficiently.

As opposed to the popular scientific computation workflow systems such as Taverna, Kepler, and Triana, which provide for building large, complex computational workflows, the LiFLow system provides only one simple workflow. However, this limitation provides an opportunity to make the LiFlow system extremely easy to use. The complicated computational workflow systems are especially useful in domains with standardized data formats and tools, such as bioinformatics (the Taverna system is specifically targeted at bioinformatic applications). However, adding new components into such workflow systems is rather difficult. In contrast, the LiFlow system is more suitable for researchers who write the simulation code by themselves. Unfortunately, due to the beta version of our project, we will be able to publish the source code later.

7 Conclusion and Future Work

The paper presents the LiFlow computational workflow system intended to automate the processing of a large number of computational experiments for living systems simulation on parallel clusters. Distinctive features of LiFlow are the automatic generation of the input data and parameters for carrying out experiment series. The system has been used for simulation of the human heart left ventricle. The use of LiFlow can significantly reduce the preparation time of a series of experiments, as well as make processing of their results more convenient.

Directions for future work include:

- Full integration with the Sumatra tool to ensure the reproducibility of launched experiments.
- Developing the mechanisms of secure integration of several computational clusters from different organizations with a single LiFLow instance in order to share computational resources and simulation results.
- Implementing the Parallel Computing System Adapters for cluster resource managers other than SLURM, as well as for cloud.
- Creating more advanced and flexible generators of experiment series integrated with GUI.

Acknowledgments. The work is supported by the RAS Presidium grant I.33P "Fundamental problems of mathematical modeling", project no. 0401-2015-0025, and by the Research Program of Ural Branch of RAS, project no. 15-7-1-26. Our study was performed using the *Uran* supercomputer of the Krasovskii Institute

of Mathematics and Mechanics and computational cluster of the Ural Federal University.

References

1. Talia, D. Workflow Systems for Science: Concepts and Tools. In: ISRN Soft Eng. 2013, 15 pages (2013). doi: 10.1155/2013/404525.
2. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T. Taverna: A tool for building and running workflows of services. In: Nucleic Acids Res. 34(Web Server issue): pp. 729-732 (2006).
3. Ludscher, B., Altintas, I., Berkley, Ch., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y. Scientific workflow management and the Kepler system: Research Articles. In: Concurrency and Computation: Practice and Experience - Workflow in Grid Systems. Vol. 18, No. 10. pp. 1039–1065 (2006).
4. Taylor, I., Shields, M., Wang, I., Harrison, A. Visual Grid Workflow in Triana. In: Journal of Grid Computing. Vol. 3, No. 3, pp. 153–169 (2005).
5. Savchenko, D.I., Radchenko, G.I. DiVTB Server: sreda vypolneniya virtual'nykh eksperi-mentov [DiVTB Server: an environment for virtual experiments executions]. In: Parallel'nye vychislitel'nye tekhnologii (PaVT'2013): trudy mezhduna-rodnoy nauchnoy konferentsii (1-5 April 2013, Chelyabinsk) [Parallel Computational Technologies (PCT'2010): Proceedings of the International Scientific Conference (1-5 April 2013, Chelyabinsk, Russia)] Chelyabinsk, Publishing of the South Ural State University. pp. 532–539 (2013).
6. Howe, B. Virtual Appliances, Cloud Computing, and Reproducible Research. In: Computing in Science & Engineering, vol. 14, no. 4, pp. 36–41 (July-Aug 2012).
7. Guo, P. CDE: A Tool for Creating Portable Experimental Software Packages. In: Computing in Science & Engineering, vol. 14, no. 4, pp. 32–35 (July-Aug 2012).
8. Davison, A.P.: Sumatra: a toolkit for reproducible research. In: Implementing reproducible research, Stodden, V., Leisch, F., Peng, R.D., pp. 57–78, CRC Press (2014).
9. Jette, M.A., Yoo, A.B., Grondona, M. SLURM: Simple Linux Utility for Resource Management. In: Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP). Vol 2862. pp.44–60 (2003).
10. Sozykin, A., Pravdin, S., Koshelev, A., Zverev, V., Ushenin, K. and Solovyova O.: LeVen – a parallel system for simulation of the heart left ventricle. In: 9th International Conference on Application of Information and Communication Technologies, AICT 2015 Proceedings, pp. 249–252 (2015).
11. Ushenin, K, Byordov, D. An HPC-Based Approach to Study Living System Computational Model Parameter Dependency. In: Proceedings of the 1st Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists Yekaterinburg, Russia. CEUR Workshop Proceedings Vol. 1513, pp. 67-74 (2015).
12. Pravdin, S.F., Dierckx, H., Katsnelson, L.B., Solovyova, O., Markhasin, V.S., Panfilov, A.V. Electrical Wave Propagation in an Anisotropic Model of the Left Ventricle Based on Analytical Description of Cardiac Architecture. In: PLoS ONE 9(5): e93617 (2014). doi:10.1371/journal.pone.0093617
13. Pravdin, S., Dierckx, H, Markhasin, V.S., and Panfilov, A.V. Drift of Scroll Wave Filaments in an Anisotropic Model of the Left Ventricle of the Human Heart. In: BioMed Research International, vol. 2015, Article ID 389830, 13 pages (2015). doi:10.1155/2015/389830.