Automatic Launch and Tracking the Computational Simulations with LiFlow and Sumatra

Evgeniy Kuklin[1,2,3] and Konstantin Ushenin[2,3]

[1] Krasovskii Institute of Mathematics and Mechanics, Yekaterinburg, Russia
[2] Institute of Immunology and Physiology of the Ural Branch of the Russian Academy of Sciences, Yekaterinburg, Russia
[3] Ural Federal University, Yekaterinburg, Russia
key@imm.uran.ru

Abstract. Reproducibility is an essential feature in the simulation of living systems. In this paper we describe the design of an automation recording system for computational simulations, which is capable of capturing both the metadata and the experimental results, and store them in an archive in a convenient form for post-processing. The complex capture process is hidden from users. The gathered environment of computational experiments allows to index and search the data about the experiments that have already been carried out. The system has been used for performing the simulation of the human heart left ventricle.

Keywords: parallel computing systems · living system simulation · data storage · computational experiment reproducibility

1 Introduction

High-performance computing simulations and large scientific experiments generate hundreds of gigabytes of data, with these data sizes growing every year. Very often, for recording the results even a standard supercomputer storage is not enough, to say nothing about the personal computer of scientists. However, keeping the results of computational experiments is essential for further reusing and postprocessing.

Another important problem that researchers in computing simulations often face is non-reproducibility of computational experiments. This problem is even more relevant in simulation of living systems and directly related to the large number of computational experiments that scientists have to carry out in order to obtain meaningful results. Many factors can affect computational results, such as a change in the version of the compiler or the required library on a supercomputer. Automated recording of experimental details and storage of simulation results will help to ensure reproducibility of the computational experiments.

Although the simulation of living systems often relies on computing clusters and supercomputers, the use of parallel computing systems requires a high degree of qualification in computer science, which many researchers involved in

living system modeling do not possess. Moreover, the data preparation for computational experiments is routine and time-consuming. We developed LiFlow [1], a lightweight workflow automation system that provides scientists a convenient graphical user interface to prepare and execute a series of computational experiments on a parallel computing system with a single click.

In this paper we propose our approach to extend the system with the process of automation of recording and storing metadata and experimental results. Details, parameters, and software environment of every experiment are automatically stored in the repository. The simulation results, obtained from experiments, are automatically copied to the archive on the dedicated storage server. These features can be achieved using Sumatra [2], an open source tool to support reproducible computational research.

The offered system is used to simulate the human heart left ventricle. However, it could also be used in other areas that require conducting computational experiments on parallel computing systems and storing their results.

2 Related Work

An urgent task is the development of the software tools for improving the reproducibility of computational experiments. Such tools 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.

One approach is based on executing experiments in a virtual environment, such as virtual machines or cloud [3]. 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. 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.

An alternative approach 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) [4]. 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 not suitable for parallel computing systems.

Should also be mentioned electronic notebooks, such as Hivebench [5]. They are well suited for accurate recording of all stages of the traditional experiments in life science, and can store some result as an image or a table. However, the amount of stored results is limited and not comparable with the volume of the

results of calculations on a supercomputer. Besides, Hivebench does not have an API, which makes challenging automatic access via scripts and negates the convenience of the computational experiment.

Thus, for our purposes the most suitable are special tools to support the reproducibility of computational experiments, which have direct access to a storage. Some of them are aimed at a particular domain, for example Madagascar [6] is used to analyze seismic data and supports multidimensional data analysis; and Sumatra is used for numerical computations. Since our work is related to modeling in the field of life science, to ensure the reproducibility of computational experiments we aimed at integrating LiFlow with Sumatra.

Sumatra aims to capture the information required to recreate the computational experiment environment instead of capturing the experimental context itself. It uses the source code of the program instead of the binaries and the general operating system configuration. Furthermore, Sumatra provides the ability to store the output data for future use in an archive. In addition, it allows to index and search the data about experiments carried out, including additional information provided by scientists. For example, if the 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 uses Sumatra for capturing and storing all information from previously conducted experiments.

3 System Architecture

The LiFlow system is designed for the simplified workflow shown in Fig. 1. During the first stage, researchers prepare the description of the experiment series, which is a set of experiments with the same model and varying parameters. The preparation includes 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 process of capturing the experimental metadata is performed. Finally, the experiments are launched on a parallel computing system. At the end of computation the simulation results are copied to the archive. Thereby, a user is only required to create a description of the experiment series, everything else is done automatically.

The LiFlow system (Fig. 2) consists of the four main components. The Computational Package Preparation Tool and the Experiment Execution GUI are installed on the researcher's personal computer, while the Experiment Execution Engine integrated with the Sumatra Module and the Experiment Catalog with the Archive 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

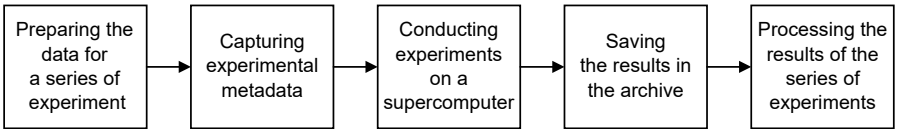

Fig. 1. LiFlow system workflow

the package to the 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. Using the Sumatra and Git commands the metadata capturing project is set up. The jobs are queued on the computational cluster using the Sumatra parallel launch options, which interacts with the resource manager of the cluster.

At the begin of the computation, Sumatra captures the environment of the computational experiment and stores it in the Experiment Catalog (database). Once the job is completed, the results of the experiment are automatically recorded by Sumatra 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 to the user.

The experiment catalog, provided by Sumatra, is used to share the initial data and the simulation results among the researchers. While preparing the series of experiments, users can browse through the results of the experiments that had been previously executed by their colleagues.

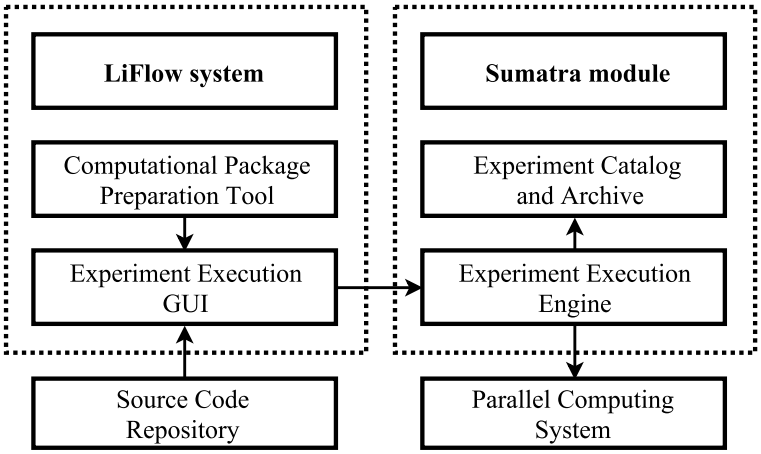

Fig. 2. LiFlow system architecture

4 Implementation Details

The computational package in the current implementation is represented by a file system directory containing 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 prepared computational package is transferred to the user's directory in the parallel computing system through the SSH protocol using the Paramiko [7] library. 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. After that, LiFlow calls the Sumatra Module script that makes the rest of the work.

First of all, the metadata capturing environment is set up. By default, Sumatra store a project information in the project local directory. In order to create the single database of all experiments of all users, we use the available `--store` record option to set the shared database file. The source code, the input files, and the launch options along with the user's description of the experiment are carefully saved and started to be publicly available. Next, the jobs are queued to performed on the supercomputer.

Sumatra can interact with the SLURM Workload Manager [8], which handles the supercomputer. So we abandon our previous implementation of setting tasks on computation and gave the control to Sumatra. Parallel computations can be performed using `--launch_mode=slurm-mpi` project option. The generated tasks for the series of experiments are placed in the SLURM job queue. After the job is complete, the data have to be transferred to a long-term storage.

Sumatra has an embedded option for compression and placing obtained data to the archive. With the `--archive` option we can set the shared folder to create the single archive for all system users. Though the original data by default is removed from the user's home directory, it is possible to disable this feature and make copies only, leaving users to do with their data whatever they want.

The shared database and archive are located at the dedicated storage server with the total capacity about 40 Tb, built on RAID arrays. This amount will be enough for several years of experimentation. As the server is located in the same network as the supercomputer, connection to it is made by the simple and reliable protocol NFSv4 and appeares as a local directory. If it is necessary to combine heterogeneous resources in different networks, the `--store` option supports public links as well. In this case transferring data to the archive has to be taken care of separately, using a mirroring data store by `--mirror` option.

Since the simulation tasks can be calculated for hours, for the convenience of users a e-mail notification system about the completion of tasks has been made. Although SLURM itself allows sending notifications about the status of the task, the project uses our own implementation of the notification based on cron and sendmail. It was done because, firstly, SLURM notifications are not very informative and can confuse users, and secondly, in the case of a series of several dozen experiments they will only lead to cluttering the mailbox.

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

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. The user needs to specify the credentials (login and password), the path to the folder with the computational package, and the email address (for job completion notification). 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. The LiFlow GUI is also written in Python using the PyQt4 library and is designed to work both on Windows and Linux.

The disadvantage of the current LiFlow implementation is the lack of the 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. 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 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. The users appreciated the convenience of the LiFlow GUI and the ability to obtain the results of simulation from the storage system. Overall, LiFlow helped the researchers from the Institute of Immunology and Physiology UrB RAS to conduct computational experiments more efficiently.

During the evaluation of the LiFLow system in the Krasovskii Institute of Mathematics and Mechanics and the Ural Federal University, we were unable to build the catalog and archive of experiments to be shared among different organizations. This problem was caused not by technical difficulties but by organizational boundaries and security considerations. However, it can be solved with the Sumatra options mentioned earlier.

Used in tandem, LiFlow and Sumatra provide the ability to combine the advantages of both systems in one solution. Sumatra is a powerful tool to support reproducible computational experiments. With its help, we can achieve the automatic capture of metadata from all running through the LiFlow system experiments, track the data received from them and automatically record the data to the archive. The useful feature is the ability of Sumatra to work with the SLURM workload manager, which simplifies job handling. At the same time the tracking tool keeps a log of all running experiments, in which any user of the system can find any experiment of his colleagues, using a date or tags. While LiFlow allows one to execute a series of computational experiments on parallel computing systems using a convenient GUI, Sumatra automatically captures

and stores the experimental environment in order to improve the experiments' reproducibility.

6 Conclusion and Future Work

The paper describes the integration LiFlow, the computational workflow system intended to automate the processing of a large number of experiments on parallel clusters, with Sumatra, which is a tool to support reproducible computational research. The system has been used for the 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.

In the future, the system can be expanded in the following areas:

- 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.
- Providing the ability to use cloud data storage for Experiment Archive.
- 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 the cluster of the Ural Federal University.

References

1. Ushenin, K.S., Kuklin, E.Y., Byordov, D.A., Sozykin, A.V.: Computational workflow system for simulation of living systems on supercomputers. In: 10th Intern. Scientific Conf. on Parallel Computing Technologies, PCT 2016; Arkhangelsk; Russia; 29-31 March 2016. CEUR Workshop Proceedings. Vol. 1576. (2016) 729–735
2. Davison, A.P., Mattioni, M., Samarkanov, D., Teleńczuk, B.: Sumatra: a toolkit for reproducible research. In: Implementing Reproducible Research. CRC Press (2014) 57–78
3. Howe, B.: Virtual Appliances, Cloud Computing, and Reproducible Research. In: Computing in Science & Engineering. Vol. 14, no. 4. (2012) 36–41
4. Guo, P.: CDE: A Tool for Creating Portable Experimental Software Packages. In: Computing in Science & Engineering. Vol. 14, no. 4. (2012) 32–35
5. Hivebench Electronic Lab notebook, https://www.hivebench.com/
6. Fomel, S., Sava, P., Vlad, I., Liu, Y., Bashkardin, V.: Madagascar: Open-source Software Project for Multidimensional Data Analysis and Reproducible Computational Experiments. In: Journal of Open Research Software. (2013) DOI: 10.5334/jors.ag
7. Paramiko: a Python implementation of SSHv2, http://www.paramiko.org/
8. 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. (2003) 44–60