Chapter 5

HPC Usability Research Team

5.1 Members

Hiroya Matsuba (Team Leader)
Motohiko Matsuda (Research Scientist)

5.2 Research Activities

The mission of the HPC Usability Team is to develop a software platform that contributes to increasing the number of uses of K computer. This team focuses on infrastructure providers, such as utility companies, as potential new users of supercomputers, expecting that simulation of infrastructure facilities will provide valuable information for its operation. For example, if a simulator of a cooling facility is available, the operator can search for an energy-efficient combination of parameters such as a temperature of cooling water, power of pumps, and positions of valves. Simulation is especially valuable because it enables operators to test parameters that are impossible to apply to the real facilities because of safety reasons.

The main obstacle in conducting facility simulations is difficulty in modeling the target facility. Complicated facilities are often composed of many components. They have to be modeled in various ways. For example, some simple parts, like a heat exchanger, may be modeled as a single dimension (1D) component. Such 1D modeling is valid if it is enough to model its overall behavior with a simple differential equation, ignoring the size, shape or structure of the parts. On the other hand, other parts may require 3D modeling with a precise modeling of the fluid flow, for example. In the worst case, the facility may contain black boxes of which even operators of the facility do not know the operating principle. In such a case, models must be created using observed data. Building models with such various modeling techniques requires deep knowledge of both mechanical and computer engineering. In addition, methods to combine the models are also needed to make a complete facility simulator from the models.

The HPC Usability Research Team is working to tackle the difficulties to create such complex simulators. To achieve this, both of the followings are important.

- Develop an easy method to make simulation models of components of facilities.
- Develop an easy method to combine different simulators.

For the easy modeling of components, we are planning to create a new domain-specific language that covers a wide variety of modeling methods such as DAE (differential algebraic equations), PDE (partial differential equations), particle simulations, agent simulations, and so on. Although different modeling method must be handled by different simulators, we still assume it possible to provide a similar user interface for different simulators. Providing different simulators with a similar user interface is valuable because it contributes to reducing the learning costs of users. For the easy combination of simulators, we are planning to develop a quick coupler. The coupler is software that picks up values from a working simulator and passes the values as input for another simulator to realize the combined simulation. Although the concept of the coupler was already introduced many year ago, it is hard to implement in a standard computer language like C/Fortran because developers have to manually specify which value of a model corresponds to which value of another model. In order to simplify this task, we aim to develop a generalized coupler that is easily applicable to all
models developed with our domain specific language mentioned above. The coupler will also connect simulators with machine learning frameworks. This is necessary when we have to develop simulation models of black box components from observed data, or when we develop an artificial intelligence that learns simulation results.

To reach the research objectives described above, we are planning to conduct our research with the following two steps. The first step is to find a particular facility to simulate and create a simulation model for that facility by ourselves. By experiencing the difficulties in creating the simulator, we, at first, develop a minimal set of a software platform that simplifies our tasks. After establishing the first platform, the second step is to find several additional targets and expand our platform so that it can be applied to the additional ones. This process enables us to develop a software platform that is both practical and general.

In this fiscal year, which is the first fiscal year of this team, we made the following progress.

- We have selected the cooling facility of K computer as our first modeling target and developed an initial version of the simulation model in collaboration with the Operations and Computer Technologies Division.
- We have invented a technique to efficiently conduct facility simulation for machine learning, especially for reinforcement learning, in collaboration with Hitachi, Ltd.

The details of these accomplishments are described in the following sections.

5.2.1 Simulation of K Computer Cooling Facilities

As the modeling target of our first example case, we selected the cooling facility of K computer. That facility is suitable as the modeling target especially because various kinds of information is available from the facility. Such information is collected by sensors installed in this fiscal year by the Operations and Computer Technologies Division (details may be reported in a later chapter of this report). Such data enables us to evaluate how accurately our simulation model reflects the actual behavior of the facility.

For overall modeling of the whole facility, we decided to use Modelica language [A]. Modelica is a domain specific language for modeling complex systems that are described with DAEs. Because Modelica is component-based language, users usually can implement models of complex systems just by connecting basic components, many of which are prepared in a standard library, with GUI.

An initial version of our model of the cooling facility is shown in Figure 5.1. This is a screenshot of a Modelica tool called Dymola [B]. Each icon in this figure is a Modelica component. Such a component is implemented either by just using a component in the standard library, by writing the Modelica language, or by connecting other components. As shown in this figure, Modelica models usually define relations between components with a few scalar values. Such modeling method is often called 1D modeling.

The 1D modeling includes coarse approximations. For example, in the model of Figure 5.1, it is assumed that cooling towers exchange heat between cooling water and the atmosphere ideally. However, in fact, because the cooling towers are located on the roof of the facility building, wind direction dramatically affects the efficiency of heat exchange.

In order to improve the accuracy of the model, we will have to make more precise models for some parts of the facility, such as the cooling tower. This case shows why we are going to develop a domain specific language that is applicable to various modeling methods. When we want some parts of a facility to be simulated more precisely than 1D modeling, we have to develop their models with different methods, such as the 3D fluid flow and heat transfer. 3D modeling is technically possible by using appropriate software like OpenFOAM, but learning both Modelica and OpenFOAM is a tedious task for model developers because they provide quite different programming models. A domain specific language that covers a wide variety of modeling method with similar user interface will help reduce such learning costs.

References


5.2.2 Efficient Execution of Simulations for Reinforcement Learning

Reinforcement learning is a machine learning technology in which a software agent learns how to react to various situations in a given environment. We expect that this technology may be used to develop artificial intelligence (AI) for facility operations. For example, by trying various options to react to an emergency situation and by leaning the resulting situation, the AI will eventually be able to take better actions.
In order to develop the AI that can take appropriate actions in such emergency situations, the AI has to have the experiences to handle the situation before the incident happens. However, it is impossible to allow AI to do trial and error in a real infrastructure facility because most infrastructure facilities should be kept in healthy and safe conditions. Simulation is used to provide a virtual environment where AI can make many mistakes and learn how to handle the situations.

We have developed a simulation system that consists of a Modelica simulator, called JModelica [C], and a machine learning framework, called ChainerRL [D]. Using this software environment, we also have developed a method to improve efficiency of simulation execution by eliminating redundant execution of simulations. This work has been done in a collaborative research project with Hitachi, Ltd. We have filed a patent regarding this efficient execution [1].

References
[C] JModelica: https://jmodelica.org/
[D] ChainerRL: https://github.com/chainer/chainerrl

5.2.3 Other Activities

We have done much engineering effort on K computer, mainly for setting up an environment for machine learning frameworks to run. We have also set up Modelica simulator on K computer.

As for the machine learning frameworks, we have set up TensorFlow [E] and Chainer [F]. Both have Python interfaces, but the core part of the frameworks is written in the C or C++ language. So, executable binaries for K computer had to be created. As for TensorFlow, many recent build tools are required, most of which is not installed on the login nodes of K computer. Therefore, we decided to cross-compile the software. By installing a GCC based cross-compiler, which can generate binary files for K computer, onto a standard x86 server with a recent Linux distribution, and by setting up the compilation tool to use the cross-compiler, SPARC executable files of TensorFlow were successfully created. Because Chainer requires only common compilation tools, we were able to compile them natively on the computing nodes of K computer using GCC.

For the Modelica simulator, we set up JModelica on K computer. JModelica also has a Python interface on the native libraries written in C/C++. We have successfully compiled this software on the computing nodes of
K computer using native GCC, although we had to manually download and install many Python modules that JModelica depends on. 

Due to lack of the functionalities of GCC to generate executable files with the HPC-ACE extension of the SPARC architecture, the performance of all the software mentioned above is not optimal for K computer.

References

[F] Chainer: https://chainer.org/

5.3 Schedule and Future Plan

We will start designing our DSL, which is the main part of our future research products, in the fiscal year 2018. We will also continue the modeling of the cooling facilities of K computer so that our simulator show some information that is beneficial for the operation of K computer.

5.4 Publications

5.4.1 Patents


5.4.2 Invited Talks


5.4.3 Presentations