WORKFLOW-BASED DISTRIBUTED ENVIRONMENT FOR LEGACY SIMULATION APPLICATIONS

Mirko Sonntag[1], Sven Hotta[1], Dimka Karastoyanova[1], David Molnar[2] and Siegfried Schmauder[2]
[1] *Institute of Architecture of Application Systems, University of Stuttgart*
Universitaetsstrasse 38, 70569 Stuttgart, Germany
{sonntag, hotta, karastoyanova}@iaas.uni-stuttgart.de
[2] *Institute of Materials Testing, Materials Science and Strength of Materials, University of Stuttgart*
Pfaffenwaldring 32, 70569 Stuttgart, Germany

Keywords: Simulation workflows, Distributed simulations, BPEL, Web services, Monte-Carlo.

Abstract: Computer simulations play an increasingly important role to explain or predict phenomena of the real world. We recognized during our work with scientific institutes that many simulation programs can be considered legacy applications with low software ergonomics, usability, and hardware support. Often there is no GUI and tedious manual tasks have to be conducted. We are convinced that the information technology and software engineering concepts can help to improve this situation to a great extent. In this poster presentation we therefore propose a concept of a simulation environment for legacy scientific applications. Core of the concept are simulation workflows that enable a distributed execution of former monolithic programs and a resource manager that steers server work load and handles data. As proof of concept we implemented a Monte-Carlo simulation of precipitations in copper-alloyed iron and tested it with real data.

1 INTRODUCTION

The importance of computer simulations increases steadily. Due to achievements in information technology it is possible to use more and more complex and hence realistic simulation models. But there is still potential to improve existing solutions. In collaborations with scientific institutes in the scope of our research project we perceived that many simulation applications are based on legacy software that was developed over years and is still in development process.

Many authors contributed to the software and may already have left the institute or organization. Usually, there is no time, money or knowledge to re-implement the tools in a modern programming language. The software is simply enhanced with new features. We experienced that there are many monolithic simulation tools in use that do not benefit from multi-core CPUs and distributed computing. The programs often cannot deal with parallel invocations, e.g. because they do not organize the result files appropriately. The manual tasks that scientists have to carry out to achieve their results are another problem. For instance, they have to create input files, copy result files between the participating applications (for calculations, visualizations, analysis, etc.), they have to start these applications, or merge results. These and other problems leave room for improvements of existing scientific simulation applications.

In this poster presentation we show a concept for a distributed simulation environment based on the workflow technology. In the last 5 to 10 years much research has been done to apply workflows for scientific applications (Deelman et al., 2004; Barga et al., 2008; Goerlach et al., 2011; Sonntag et al., 2010). We are convinced that workflows possess great potential to improve the tool support for many scientists. These are the tools scientists work with every day. There is a need to automate and optimize simulation processes, to exploit recent developments in hard- and software, and to improve user-friendliness and flexibility of simulation applications. This can be done if modern IT and software engineering technologies and techniques are used. At the same time, scientists do not want to re-write existing code or re-implement applications for simulation tasks. The proposed concept addresses these and other requirements.

Sonntag M., Hotta S., Karastoyanova D., Molnar D. and Schmauder S..
WORKFLOW-BASED DISTRIBUTED ENVIRONMENT FOR LEGACY SIMULATION APPLICATIONS.
DOI: 10.5220/0003444400910094
In *Proceedings of the 6th International Conference on Software and Database Technologies* (ICSOFT-2011), pages 91-94
ISBN: 978-989-8425-76-8

The paper is structured as follows. Section 2 presents the use case for the simulation of solids. Section 3 describes the infrastructure for the distributed simulation applied to this use case. Section 4 closes the paper with a conclusion.

2 USE CASE: SIMULATION OF SOLID BODIES

The macroscopic mechanical properties of metals strongly depend on their underlying atomistic structures. Copper-alloyed α-iron, e. g., changes its material behaviour during the ageing process, especially when operated at higher temperatures of above 300°C. In that case, precipitates form within the iron matrix, yielding to precipitation strengthening of the material (Kizler et al., 2000) followed by a decrease of the material strength as the precipitates grow further in time. This complex behaviour is modelled with the help of a Kinetic Monte-Carlo (KMC) approach (Schmauder and Binkele, 2002) based on a code developed by Soisson et al. (1996).

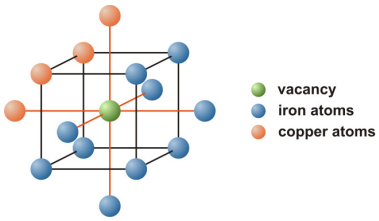

Figure 1: Body-centered cubic crystal lattice with a vacancy-site and neighbour atoms for one possible atom configuration.

As starting configuration, a number of copper atoms are placed into a fixed body-centered cubic iron lattice by exchanging an iron with a copper atom. A vacancy within the simulation box allows the movement of an atom by site exchange between the vacancy and a neighbouring atom (Fe, Cu) (Fig. 1). In every Monte Carlo (MC) step, the jump frequencies for each of the neighbours of the vacancy are calculated. By applying a rejection-free residence time algorithm, one of these possibilities is selected as the new position of the vacancy. In the long run, a series of vacancy jumps during the simulation yields the formation of precipitates with mean radii above 1 nm (Fig. 2).

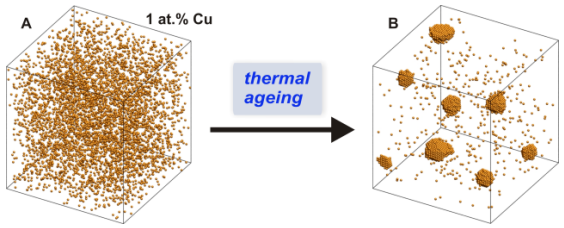

Figure 2: KMC simulation of precipitation. Iron atoms are transparent for better visualization. Start configuration with randomly distributed copper atoms (A). Precipitates form during the thermal ageing process (B) (Molnar et al., 2010).

At desired MC time steps the atom configuration (i.e. a snapshot) is analyzed yielding particle size distributions, mean radii of particles and the degree of supersaturation of the remaining solid solution.

The existing simulation application *opal* (Binkele and Schmauder, 2003) consists of five monolithic Fortran programs where the individual programs require manually created input data and are started manually. Two of them build up the starting configuration and calculate the input values (e.g. the energies). In addition to iron and copper, two more atom species can be incorporated. The precipitation process is simulated by the main opal application. After specified time intervals, snapshots are generated. The analysis of these snapshots is performed by the last two programs which identify the precipitations within the matrix and determine size distributions and mean radii. Finally, the results are visualized, e.g. with MatLab or Gnuplot. All in all, the overall simulation can be subdivided into four phases: preparation, simulation, analysis and visualization.

3 WORKFLOW-BASED DISTRIBUTED SIMULATIONS

The workflow technology (Leymann and Roller, 2000) can improve the setup and execution of scientific simulations. With the help of workflows scientists can adapt the simulation logic without touching the simulation code (e.g. use another visualization program). Fault handling features of workflows languages increase the robustness of simulation applications, especially if the simulation functions are implemented with a language that offers only restricted fault handling mechanisms. The monitoring component of workflow engines enables users to observe the progress of their workflows, i.e. their simulations. BPEL and other workflow languages enable the invocation of

programs in a network using Web services. Distributed applications can thus easily be implemented by orchestrating functions located on different machines. Further, different application types can be integrated and manual steps can be automated, e.g. the invocation of a visualization tool after a simulation run is finished.

The conceptual architecture for a distributed simulation environment (Fig. 3) shows the components and their dependencies needed for an application of workflows in the scientific domain.

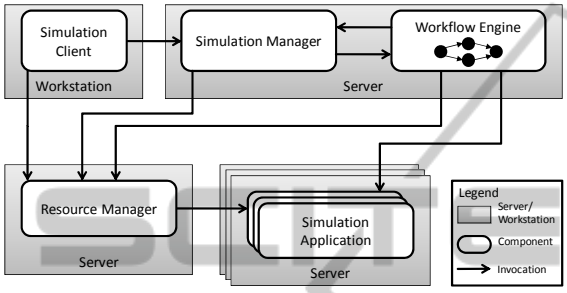

Figure 3: Architecture of the simulation environment.

The *simulation applications* are the domain specific part of the software. A thin wrapper publishes the programs in a network so that they can be invoked and accessed remotely. For the use case, we extended the interfaces of the five Fortran applications to allow their invocation with parameters. Furthermore, we created a WS wrapper for them. Since there is no WS toolkit for Fortran programs, we built a Java wrapper that invokes the Fortran executables.

The *resource manager* (RM) is a simulation application registry, data storage and supervisor of the simulation infrastructure. Participating servers and simulation applications are registered at the RM prior to or during the execution of simulations. The RM can even install software on demand on an idle server with the help of provisioning techniques as used in Grid or Cloud computing (Foster and Kesselman, 2004). Clients that want to use a simulation application have to ask the RM for permission. Permission is granted with the help of service tickets that warrant exclusive usage right. This is comparable to advanced reservation techniques in Grids (Foster and Kesselman, 2004). The goal of the RM is to ensure load balancing. Furthermore, it checks the availability of simulation servers to detect network partitions. Clients get informed so that they can react accordingly (e.g. by requesting the service again). Finally, the resource manager can be used as storage for simulation data (e.g. simulation parameters, configuration, results). Each simulation run gets its own simulation context where data can be saved, organized and deleted. The context correlates data items that belong to the same simulation run. This improves reproduction of simulations because the configurations and input data as well as result data are assembled and can be observed by scientists at a glance. The data items do not have to be searched for and correlated later on, which may not be possible at all since simulation applications may not have a sophisticated storage mechanism (e.g. legacy simulation applications might overwrite simulation data of former runs).

The *workflow engine* is responsible for executing the simulation workflows that implement the logic of the simulations. We implemented the use case with two BPEL workflows. The main workflow acquires a service for the MC simulation, executes the simulation, collects the results and starts the post-processing workflow for each intermediate result. The post-processing workflow acquires and invokes the services for calculating the precipitations and their radii. If post-processing is done, the main workflow invokes Gnuplot to visualize the results.

The *simulation manager* (SM) offers domain-specific functions to configure and start simulations and to access simulation context data. It makes use of the resource manager as storage facility. In multi-scale or multi-physics simulations a single simulation can consist of multiple workflows. The SM can be used to manage these different participating workflows.

The *simulation client* is a GUI used by scientists to interact with the simulation environment. It provides the domain-specific functions of the SM and the general functions of the RM to the scientists. For the use case we have additionally implemented functions to store input data as profile for other simulation runs, a file explorer that shows the context of simulation runs, and a 3D visualization of the intermediate lattice snapshots (Fig. 4). The latter allows observing the convergence of the simulation results during run time.

For monitoring purposes we have implemented a view that shows all service ticket requests that are sent to the resource manager and a view that lists all issued service tickets. This information enables scientists to observe the status of the simulation environment and existing problems (e.g. a long list of open service requests may indicate too few available instances of a simulation service in the system).

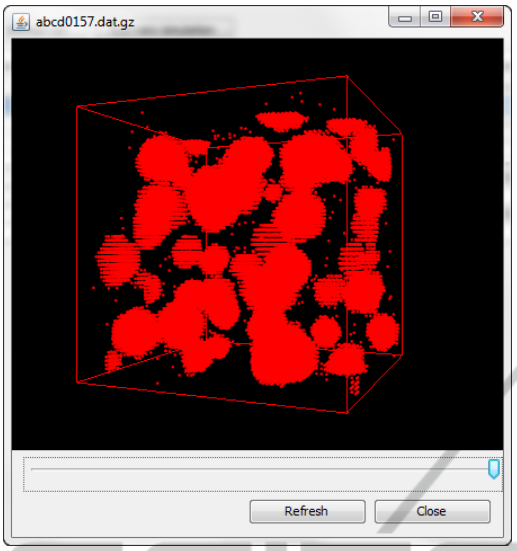

Figure 4: 3D preview of the simulation snapshots (Hotta, 2010).

4 CONCLUSIONS

In this paper we have presented a concept for an infrastructure that enables the distributed execution of scientific simulations. Core of the infrastructure are simulation workflows that reflect the logic of simulations and a resource manager that controls the work distribution on the participating machines and that works as storage for simulation data. The infrastructure addresses main requirements of scientists on a simulation environment and hence can improve the tool support for scientific simulations, e.g. to automate manual tasks, to enable distributed execution of legacy software.

As a proof of concept we implemented an MC simulation of solid bodies based on BPEL and tested it with realistic data. The software improves the simulation process and the former application to a great extent. The scientists now have a GUI to start and monitor their simulations, some of the manual steps could be automated (e.g. start of post-processing and visualization of results), and multiple CPU cores and distributed computing can be exploited. We are convinced that our consideration can help scientists with their every day work.

ACKNOWLEDGEMENTS

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart. We thank Peter Binkele who contributed the MC simulation code opal to our work.

REFERENCES

Barga, R., Jackson, J., Araujo, N. et al., 2008. The Trident Scientific Workflow Workbench. In *Proc. of the IEEE International Conference on eScience*.

Binkele, P., Schmauder, S., 2003. An atomistic Monte Carlo simulation for precipitation in a binary system. In *International Journal for Materials Research, 94*, pp. 1-6.

Deelman E., Blythe, J., Gil, Y. et al., 2004. Pegasus: Mapping Scientific Workflows Onto The Grid. In *Proceedings of the 2nd European AcrossGrids Conference*, pp. 11-20, Springer-Verlag.

Foster, I., Kesselman, C., 2004. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition.

Goerlach, K., Sonntag, M., Karastoyanova, D. et al., 2011. Conventional Workflow Technology for Scientific Simulation. In: Yang, X., Wang, L., Jie, W., 2011. *Guide to e-science*. Springer-Verlag.

Hotta, S., 2010. Ausführung von Festkörpersimulationen auf Basis der Workflow Technologie. *Diploma Thesis No. 3029*, University of Stuttgart.

Kizler, P., Uhlmann, D., Schmauder, S., 2000. Linking Nanoscale and Macroscale: Calculation of the Change in Crack Growth Resistance of Steels with Different States of Cu Precipitation Using a Modification of Stress-strain Curves Owing to Dislocation Theory. In *Nuclear Engineering and Design, 196*, pp. 175-183.

Leymann, F., Roller, D., 2000. *Production Workflow – Concepts and Techniques*. Prentice Hall.

Molnar, D., Binkele, P., Hocker, S., Schmauder, S., 2010. Multiscale Modelling of Nano Tensile Tests for Different Cu-precipitation States in α-Fe. In: *Proceedings of the 5th International Conference on Multiscale Materials Modelling*, pp. 235-239, Fraunhofer Verlag.

Schmauder, S., Binkele, P., 2002. Atomistic Computer Simulation of the Formation of Cu-Precipitates in Steels. *Computational Materials Science, 24*, pp. 42-53.

Soisson, F., Barbu, A., Martin, G., 1996. Monte-Carlo Simulations of Copper Precipitates in Dilute Iron-Copper Alloys During Thermal Ageing and Under Electron Irradiation. In *Acta Materialia, 44*, pp. 3789-3800.

Sonntag, M., Karastoyanova, D., Deelman, E., 2010. Bridging the Gap Between Business and Scientific Workflows. In *6th IEEE International Conference on e-Science*.