Using HLA and Grid for Distributed Multiscale Simulations

Katarzyna Rycerz[1], Marian Bubak[1,2], Peter M.A. Sloot [2]

[1]Institute of Computer Science, AGH, al. Mickiewicza 30,30-059 Kraków, Poland
[2]Faculty of Sciences, Section of Computational Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{kzajac | bubak }@agh.edu.pl, sloot@science.uva.nl
phone: (+48 12) 617 39 64, *fax:* (+48 12) 633 80 54

Abstract. Combining simulations of different scale in one application is non-trivial issue. This paper proposes solution that supports complex time interactions that can appear between elements of such applications. We show that High Level Architecture, especially its time management service can be efficiently used to distribute and communicate multiscale components. Grid HLA Management System (which was presented in our previous work [10]) is used to run HLA–based distributed simulation system on the Grid. The example application is build from simulation modules taken from Multiscale Multiphysics Scientific Environment (MUSE)[8], which is sequential simulation system designed for calculating behaviour of dense stellar systems like globular clusters and galactic nuclei.

Keywords: multiscale simulation, Grid computing, HLA, distributed simulation

1 Introduction

Multiscale simulations is a very important and interesting field of research. Examples include approach to create multiphysics model of capillary growth [11] or modeling colloidal dynamics [3]. Another example is Multiscale Multiphysics Scientific Environment (MUSE)[8] for simulating dense stellar systems like globular clusters and galactic nuclei. The MUSE currently consists of the Python scheduler and three simulation modules of different time scale: stellar evolution (in macro scale), stellar dynamics (nbody simulation - in meso scale) and hydro dynamics (simulation of collisions - in micro scale).

Combining simulations of different scale in one application is a complex and non-trivial issue [4, 5]. In particular, it requires advanced and flexible time management techniques (e.g. ability of joining together simulations of different internal time management). From that point of view, it would be useful to adapt one of the existing solutions suited for distributed interactive simulations that can fulfill this requirement. One of the important standards is High Level Architecture (HLA) [6] that provides various services needed for this kind of applications

such as time management with the ability to connect time-driven and event-driven as well as optimistic and conservative simulations together. It also takes care of data distribution management and allows all application components to see the entire application data space in an efficient way. There are many implementations of open source HLA standard as well as its closed source [12]. In this paper we show, how multiscale simulation can benefit from HLA time management services. As example, we use simulation modules from MUSE environment.

Another important aspect worth to be addressed is the ability of flexible and transparent creation of distributed multiscale simulation system according to user needs. This issue include requirement of interoperability, composability and reusability of exiting models created by other researchers. For this purpose we propose to use Grid technology as it is oriented towards joining geographically distributed communities of scientists working on similar problems - this will allow users working on multiscale simulations to more easily exchange the models already created. Therefore, the attempt to integrate HLA with new possibilities given by Grid is a promising approach useful for multiscale simulations with time management supported by HLA. As there is already much work on improvements of HLA functionality regarding conditions in Grid environment (a good example is the implementation based on Grid Services [9]), we are not planning to develop our own implementation. Instead, we would like to show, how using HLA on the Grid can be beneficial for multiscale simulations. To run the simulation we are using Grid HLA Management System (G-HLAM) described in [10], where you can also find detailed analysis of challenges of integrating HLA with the Grid. In the future, we also plan to use component approach to extend support for composability of simulation models.

This paper is organized as follows: in Section 2 we briefly describe HLA time management, in Section 3 we analyze the typical time interactions between multiscale components basing on MUSE modules [8]. In Section 4 we describe our first attempts of running HLA–based distributed multiscale simulation on the Grid and the performance results. In Section 5 we describe conclusions and plans for future work.

2 Overview of time management in HLA

The High Level Architecture (HLA) standard [6] defines an infrastructure for developing distributed interactive simulations. In HLA terminology each component of a distributed simulation is called a federate and can form federation with other federates. In HLA, time management is concerned with the mechanisms for controlling the advancement of each federate along the federation time axis. The perception of the current time may differ among participating federates. So called *regulating federates* regulate the progress in time of federates that are designated as *constrained*. A federation may be comprised of federates with any combination of time management models. Regulating federates are able to send events or data objects with time stamps and constrained federates are able to receive such objects in time stamp order. The maximal point in time which

the constrained federate can reach at certain moment is calculated dynamically according to the position of regulating federate on the time axis.

The constrained federate can ask HLA runtime infrastructure to proceed to to the next point in time calculated by adding time step to the current time (for time–driven simulations) or calculated as the time of the next event received (for event driven simulations). Additionally, optimistic federates can also ask to receive all of the events that have been sent in the federation execution regardless of the time-stamp ordering. The messages that are received with a time-stamp less than messages already sent may invalidate the previous messages. In this case, HLA provides retraction mechanism that can be used.

3 HLA support for types of time interactions in multiscale simulations

We have adapted three simulation modules of different time scale taken from MUSE [8]: stellar evolution (in macro scale), stellar dynamics (nbody simulation - in meso scale) and hydro dynamics (simulation of collisions - in micro scale).

The main multiscale simulation elements are shown in the Fig 1. It shows two steps of evolution and four steps of dynamics (the number of steps is chosen for simplicity - actually there are more steps of dynamics within time of one evolution step). Simulation of collisions is seen by evolution and dynamics as a point in time. Collision is triggered by dynamics and data are sent from collision to both dynamics and evolution. Apart from that, evolution sends data to dynamics. We identified three needed types of time interactions between multiscale elements.

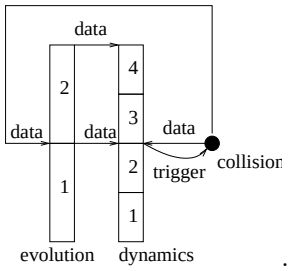

Fig. 1. Multiscale simulation elements and their interactions

Meso scale triggers micro scale and waits for the results. In simplest case (shown in the Fig.2) simulation of stellar dynamics (meso scale) can detect the situation when two stars become close. Then, the collision simulation (micro scale) should be performed (triggered). As the collision takes time in smaller scale then dynamics and the computed data is sent from micro scale to meso scale, the dynamics should wait as the collision finishes.

Macro scale and meso scale running concurrently - conservative simulation. In this case (shown in the Fig.3), evolution (macro scale) and dynamics (meso scale) can run concurrently. The data are sent from macro scale to meso scale as star evolution data (change of star mass) is needed by dynamics. No data is needed from dynamics to evolution . Single step of dynamics simulation is shorter in terms of simulation time units than that in evolution, so it needs to take more steps to reach the same point of time. Also, complexity of the single

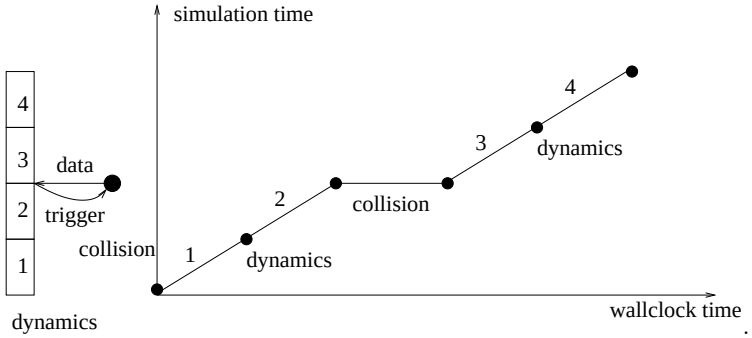

Fig. 2. Interaction between dynamics and collision

step of dynamics (and related execution time) is greater then this of evolution. Therefore, there is less probable that dynamics simulation will frequently wait for its data until evolution reaches next step. As shown in Fig.3, point A1 is earlier on wallclock time axis than point B1 (for simplicity in the Fig.3 steps of both simulations are equal. In real experiment, which is described later in Section 4, dynamics part have to calculate above 1000 simulation steps to get to point B1, what can take even 25 seconds of wallclock time, while evolution performs one step to get to point A1 in 1.1 milisecond.) Therefore, we propose the conservative type of interaction between these two simulations as it allows to run two simulations concurrently and should not generate frequent delays idle time of waiting simulation. Also, in optimistic solution the whole state of dynamics would have to be rolled back in case the evolution was late as data sent by evolution impact the whole dynamics simulation.
The simulation system has to make sure that dynamics will get update from

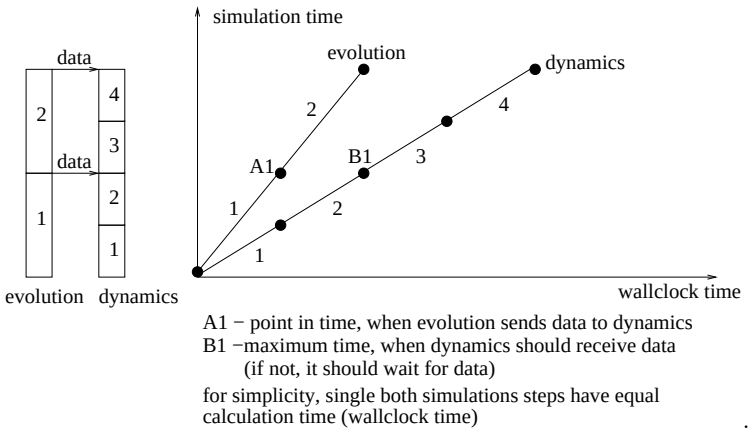

Fig. 3. Interaction between evolution and dynamics

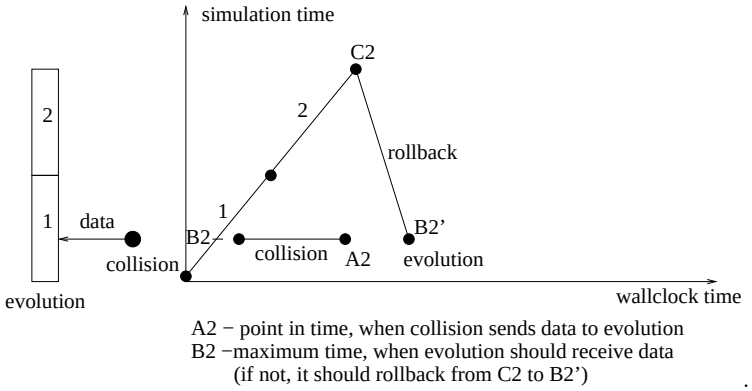

Fig. 4. Interaction between evolution and collision

evolution before it actually passes the appropriate point in time. The time management mechanism of *regulating* federate (evolution) that controls time flow in *constrained* federate (dynamics) could be there very useful. The maximal point in time which the constrained federate can reach at certain moment is calculated dynamically according to the position of regulating federate on the time axis.

Macro scale and micro scale running concurrently - optimistic simulation. In this case (Fig.4), the collisions simulation (micro scale) impact part of evolution data (macro scale). The start time of collision is independent on evolution. Collision means that the set of simulated stars changes (two stars disappear, and one star of different mass appears in their place). In the case the evolution already passed the point of time of collision (e.g. in the Fig.4 point A2 is later then B2 on the simulation time axis), it has to rollback (in the Fig.4 the evolution has to rollback from point C2 to B2')

As the data from micro scale impacts only part of macro scale simulation (the evolution of each star is calculated independently of other stars), the optimistic simulation can be a reasonable solution. HLA time management [6] offers also support for this kind of simulations - e.g. ability to check messages with future time stamps and retract messages that were sent basing on the future data.

Interaction summary. All presented types of time interactions have different features. In the first and second interaction, the data flow effects whole receiving simulation, while in third case only a part (evolution calculates every star separately, so data from collision does not have effect on calculations of not colliding stars). The first type of interaction is sequential (dynamics triggers new collisions and waits for the results), while the other two are parallel: second one is conservative and third one is optimistic. In HLA all forms of time management may be linked together and conservative simulations can interact with optimistic simulations. Therefore, it allows to link three scenarios of time interaction described above in the natural way and is a good choice for this kind of complex simulation system.

4 HLA–based distributed MUSE on the Grid

G-HLAM. In our previous work, we developed G-HLAM system that allows for efficient execution of HLA–based applications on the Grid [10]. One of the main goals was to support HLA legacy applications, so the actual communication between simulation elements (federates) are done through HLA communication bus. However, management of the application (performance monitoring and migration) is done on the Grid Services level. The group of main G-HLAM services consists of a *Broker Service* which coordinates management of the simulation, a *Performance Decision Service* which decides when the performance of any of the federates is not satisfactory and therefore migration is required, and a *Registry Service* which stores information about the location of local services. On each Grid site supporting HLA there are local services for performing migration commands on behalf of the *Broker Service* as well as for monitoring of federates and benchmarking. The *HLA–Speaking Service* is one of the local services interfacing federates running on its site to the G-HLAM system.

Performance results. We have used G-HLAM for running HLA–based distributed multiscale application on the Grid. For this experiment we have chosen two modules *dynamics* and *evolution* to show second type of time interaction described in Section 3 (macro scale and meso scale running concurrently - conservative simulation). This experiment is good example of using HLA time management features, when one simulation controls time of the other. In our experiment, we compared performance of this modules when running them firstly in legacy MUSE environment and secondly as HLA–based distributed simulation on the Grid. The experiments were performed on DAS2: legacy MUSE and

action	average, sec	σ, sec
Actions independent on simulation steps number (HLA–based distributed multiscale on the Grid)		
job submission (GRAM and local job manager)	15.2	1
HLA initial actions	7.3	1.1
HLA quit actions	0.07	1.1
Actions dependent on simulation steps number (Legacy MUSE)		
dynamics	45.8	0.1
evolution	0.001	0.00005
update from evolution to dynamics	0.061	0.001
total loop time	45.9	0.2
Actions dependent of simulation steps number (HLA–based distributed multiscale)		
dynamics	49	3
synchronization with evolution	0.016	0.001
total loop time	49	3

Table 1. Comparison of time of actions for legacy MUSE and for HLA–based multiscale simulation on the Grid (sum of 10 steps)

dynamics part of HLA application was run at grid node at Leiden, evolution part of HLA application was run at Delft and RTIexec HLA control process was run at Amsterdam. All grid nodes have the same architecture (clusters of two 1-GHz Pentium-IIIs nodes connected with internal Myrinet-2000 network). Fast ethernet (10Gbps) is used as the external network between Grid nodes. The application setup and actual submission of HLA components was done by G-HLAM (with Globus Toolkit 3.2 and HLA RTI-1.3NGv5). We have calculated average values from 10 runs. The actions independent of number of simulation steps (time of submission, HLA setup and finalizing functions) are shown in the upper part of Tab.1. The rest of Tab.1 shows execution time of actual simulation loops. The experiments were done on 100 star system (first 10 simulation steps). The middle part of Tab.1 shows performance results of legacy MUSE environment. It is sequential execution consisting of: running dynamics, running evolution and updating dynamics from evolution in each step. The lower part of Tab.1 shows the experiment results with the same modules, but running concurrently on the Grid and using HLA time management. As we observed, execution time of evolution is much shorter then dynamics, so it does not delay its execution (although dynamics is controlled by evolution as described in Subsection 3). As both modules were running concurrently and evolution part finished quicker, we have shown results of the longer part (dynamics). The results include actual calculation time (which are similar to legacy execution) and synchronization time (receiving data from evolution, getting permission to advance simulation time). It should be noted that synchronization (which in fact is the most important overhead of HLA distribution as it is repeated in the loop) does not take much time, because evolution is quicker and sends data in advance, so dynamics has only to process what was delivered before. However, as the evolution execution time is much shorter then dynamics, the results of legacy and distributed simulations are comparable as the dynamics time totally dominates sequential execution. We can draw a conclusion that the distributed solution would be beneficial for modules that are both significant for sequential execution time, but with regulating module (in our case evolution) quicker then constrained module (in our case dynamics).

5 Conclusions and future work

In this paper we have shown how multiscale simulations can benefit from HLA, especially its time management facility. We have described and analyzed the typical the time interactions between multiscale components basing on modules taken from MUSE [8], which originally is sequential simulation environment designed for dense stellar systems. For the experiment we have chosen one of the described types of time interaction as it is the most typical example of using HLA time management features, when one simulation controls time of the other. However, we plan to add also other interaction types in the near future. We have shown how the multiscale components can be distributed using HLA and benefit from it. G-HLAM system was used to run the simulation on the Grid

which allows for better usage of available resources needed by such applications. The performance results have shown that overhead of HLA–based distribution (especially its repeating part of synchronization between multiscale elements) on the Grid is small and can be beneficial for multiscale simulations.

The results presented in this paper are a good starting point for building Grid component framework for HLA–based simulations that will support a user in dynamic set up of multiscale simulation system comprised of chosen components.
Acknowledgments The authors wish to thank Simon Portegies Zwart for valuable discussions on MUSE and Maciej Malawski for discussions on component models. The support from Polish Foundation for Science (FNP) is acknowledged. This research was also partly funded EU IST Project CoreGRID and the Polish State Committee for Scientific Research SPUB-M.

References

1. R. Armstrong et al. The CCA component model for high-performance scientific computing. *Concurr. Comput. : Pract. Exper.*, 18(2):215–229, 2006
2. X. Chen, W. Cai, S. J. Turner, Y. Wang: SOAr-DSGrid: Service-Oriented Architecture for Distributed Simulation on the Grid. Principles of Advanced and Distributed Simulation (PADS) 2006: 65-73
3. W. Dzwinel, D.A. Yuen, K. Boryczko, Bridging diverse physical scales with the discrete-particle paradigm in modeling colloidal dynamics with mesoscopic features, Chemical Engineering Sci., 61, 2169-2185, 2006
4. A. G. Hoekstra, E. Lorenz, J-L. Falcone, B. Chopard, Towards a Complex Automata Framework for Multi-Scale Modeling: Formalism and the Scale Separation Map, in Y. Shi et al. (Eds.): ICCS 2007, Part I, Lecture Notes in Computer Science 4487 (Springer), pp. 922-930, 2007.
5. A.G. Hoekstra, S. Portegies Zwart, M. Bubak, and P.M.A. Sloot, Towards Distributed Petascale Computing, in D. Bader (Ed.) Petascale, Computing: Algorithms and Applications, Chapman & Hall / CRC Press, Taylor and Francis Group (expected publication december 2007)
6. High Level Architecture specification - IEEE 1516
7. M. Malawski, M. Bubak, M. Placek, D. Kurzyniec and V. Sunderam: Experiments with distributed component computing across grid boundaries. In *Proceedings of the HPC-GECO/CompFrame workshop in conjunction with HPDC 2006*, Paris, France, 2006
8. MUSE Web page http://muse.li/
9. K. Pan, S. J, Turner, W. Cai and Z. Li: A Service Oriented HLA RTI on the Grid accepted by Principles of Advanced and Distributed Simulation (PADS) 2007
10. K. Rycerz, Grid-based HLA Simulation Support. PhD thesis, University of Amsterdam, June 2006, promoter P.M.A. Sloot, copromoter: M.Bubak.
11. D. Szczerba, G. Székely, H. Kurz: A Multiphysics Model of Capillary Growth and Remodeling. in: in: V. N. Alexandrow, G. D. van Albada, P. M. A. Sloot, J. Dongarra (Eds.), Proceedings of Computational Science - ICCS 2006, 6th International Conference Reading, UK, May 2006, volume II, LNCS 3992, Springer, 2006, pp. 86-93
12. Wikipedia - list of HLA implementations http://pl.wikipedia.org/wiki/High_Level_Architecture