Vitamins : Visual and In Situ Analytics for Molecular Interactive Simulation

Marc Baaden, Matthieu Dreher, Nicolas Ferey, Sébastien Limet, Jessica Prevoteau-Jonquet, Bruno Raffin, Sophie Robert, Mikael Trellet

VItAMInS : Visual and In Situ Analytics for Molecular Interactive Simulation

Marc Baaden*, Matthieu Dreher†, Nicolas Ferey‡, Sébastien Limet§,
Jessica Prevoteau-Jonquet*, Bruno Raffin†, Sophie Robert§ and Mikael Trellet‡
*Laboratoire de Biochimie Théorique, CNRS, UPR9080,
Univ. Paris Diderot, Sorbonne Paris Cité
13 rue Pierre et Marie Curie, 75005 Paris, France
†INRIA, LIG,
Grenoble, France
‡ LIMSI, CNRS, UPR3251,
B.P. 133 91403 Orsay cedex, France
§Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022,
F-45067 Orléans, France

System Demonstration

I. Introduction

Molecular dynamics simulations are now classically used to study the dynamics of biological complexes in structural biology. The efficiency of simulation tools such as NAMD [6] or Gromacs [4] allows to study very complex biological objects. Such simulations produce huge amounts of data. New ways to explore and analyze these data need to be defined in order to help the scientist to extract the relevant biological information in these systems. Such applications both require intensive computing and data processing.

Two ways are possible to address the difficulty to explore and interpret the generated data. *Post mortem* analyses are based on results saved as trajectories which can be post processed and complemented with further analysis. Since the analysis is *post mortem*, it is possible to explore the data as many times as one needs, incrementally adding new analysis, but it is not possible to modify the conditions of the simulation. *In situ* and *in transit* live analysis consist in data processing as they are generated. In these cases, a live feedback can be associated to steered techniques to change the running simulation in modifying its parameters for example. It is also interesting to optimize the use of HPC computers by using all its computing resources during the simulation and also avoiding the generation of uninteresting data.

Both *live* and *post mortem* analyses require to build modular applications since they usually rely on very specialized pre-existing software such as for scientific visualization, physical or biological analysis etc. The coordination of the different parts of the application needs to be carefully done in order to maintain the simulation performance and cope with the data size.

VItAMInS is a framework that relies on the component-based middleware FlowVR [1] and allows to develop both *in situ* and *post mortem* applications for biological system analysis. The objectives of the demo is to show how VItAMInS can facilitate the construction of such an application and make it flexible to add new modules or to use it on different platforms (personal computers, HPC clusters, virtual reality environments etc.). An example of *in situ* and *in transit* processing allowing efficient *live* analysis will be described and an example of *post mortem* trajectory analysis will be demonstrated.

II. Overview

FlowVR is a component-based middleware providing the means to develop and run high performance interactive applications for scientific visualization on distributed architectures. Its key idea is the following: each component, called *module*, executes an iterative loop which consists in waiting for input data, consuming them and producing some output data. The module does not care how the inputs are generated nor how its outputs will be used. The dataflow network between the modules of a FlowVR application manages both data dispatching and synchronization policy between the distributed modules. FlowVR is able to handle complicated communication schemes such as broadcasts, scattering or gathering data. It can filter or transform data from one module to another to reduce communications between modules. FlowVR is also able to avoid buffer overflows thanks to synchronization schemes based for instance on greedy communications controlled by ending signals emitting by modules. These schemes make the coordination of the global application totally decentralized which is one of the keys of FlowVR efficiency.

A special process called FlowVR daemon runs on each node hosting a module and is in charge of message exchanges

between modules. Each daemon has the knowledge of the running application and more precisely of the communication schemes. This way, there is a clear separation of concerns: the modules are in charge of processing tasks and FlowVR daemons filter and route the messages between modules. Thus FlowVR modules can be used in any application without being recompiled and any module may be replaced by any other module with the same interfaces (input and output ports) in an application.

The VItAMInS framework proposes some predefined modules and communication schemes that can be used to assemble either an *in situ* or a *post mortem* analysis application. Thanks to FlowVR flexibility, VItAMInS can be enriched with new modules as well as new communication schemes. Here are some of the predefined modules:

a) Gromacs: This module is a patched version of the well known molecular dynamics software package Gromacs [4] which transforms it into a FlowVR module. This module has been designed to include *in situ* and *in transit* features [3] reducing data gathering on big simulations. Some experiments [3] showed the good performance and the scalability of this module on simulations containing several hundreds of thousands of atoms processed on about two hundred cores.

b) Hyperballs: This module is a molecule viewer [2] based on sphere and hyperball representations of atoms and their links. Thanks to an optimized use of graphic shaders, it is possible to display different representations of molecules (e.g. van der Waals, licorice or ball and stick) in a uniform and very efficient way. Hyperballs is able to display interactively (i.e. allowing the user to navigate fluently in the scene) molecules containing more than 500 000 atoms.

c) TrajReader and TrajPlayer: These two modules are mainly use in *post mortem* analysis. TrajReader is in charge of accessing the different frames of a trajectory file produced by a molecular simulation. This access is optimized with an indexing process of the frames to navigate fluently in the trajectory. TrajPlayer is a graphical interface that resembles a DVD player where it is possible to go forward and backward, to accelerate, slow down or pause the reading. It is possible to loop on a selected part of the trajectory. This module mainly controls the behavior of the trajectory reader.

d) MDAnalysis scripts: MDAnalysis [5] is a Python toolkit that provides various classical algorithms used to analyse some properties on the structure and the dynamics of the molecular system. VItAMInS includes a framework to write MDAnalysis scripts that are FlowVR modules. These modules can be connected to the TrajReader to perform its analysis.

e) GraberStats: This module relies on the Qwt library and can display graphs that represent graphically the results of the MDAnalysis scripts. These graphs are very useful to provide additional information on the running simulation displayed by the molecule viewer in order to better analyse it.

f) Interaction modules: This set of modules is used for human interactions with the visualization to navigate in the molecule. It permits to select some atoms in it with various peripherals such as a 3D mouse, joysticks etc. In the case of *in situ* analysis it is possible to use interactions to apply new forces on some atoms to drive the simulation to a particular state. VItAMInS includes some modules to help the user to navigate in the scene using structural informations on the displayed molecules [7].

The description of the modules shows the capacity of VItAMInS to integrate very heterogeneous codes in a single application. The VItAMInS application can be enriched by new modules that could be legacy codes patched to be transformed into FlowVR modules. The way a FlowVR application is launched allows to run the same application on very different platforms ranging from a personal computer to a cluster of several thousand cores.

Building a VItAMInS application consists in writing a Python script that describes the modules composing the application and the FlowVR network that connects them. This network is based on communications between module interfaces. FlowVR offers some predefined communication schemes including classical broadcast, scatter or gather patterns to deal with communication from or to parallel modules. It offers synchronization schemes to filter data in order to avoid buffer overflows. These filters are daemon plugins and it is possible to specify its own filters.

In the next section, an example of a *post mortem* VItAMInS application is given to illustrate concepts introduced above and to describe the demo.

III. VITAMINS DEMO

Figure 2 illustrates a simplified *post mortem* analysis application based on *TrajReader* and *TrajPlayer* modules. In this application, the *TrajReader* module reads the atom positions of each frame of the stored trajectory and delivers them to the *hyperballs* module. It is controlled by the *TrajPlayer* interface that gives it orders indicating the way the frame needs to be read. The communication with the *hyperballs* module is a FIFO one. However, as the *TrajReader* frequency is very high compared to the *hyperballs* frequency, the *TrajReader* iterative loop consists in waiting for orders from *TrajPlayer* but also for the ending signal of *hyperballs*.

The application uses a space navigator device to interact with the visualization. The *Space3D* and the *DeviceManager* modules are FlowVR interaction modules available in VItAMInS . *Space3D* delivers some analog events that *DeviceManager* translates in camera motions. The camera motions are sent to *Hyperballs* and allow to navigate in the scene. As the *Space3D* frequency is very high, the camera motions delivered by *DeviceManager* could generate a buffer overflow on the *camera* interface of *Hyperballs*. To avoid this overflow, the communication is based on a *greedy* filter which keeps only

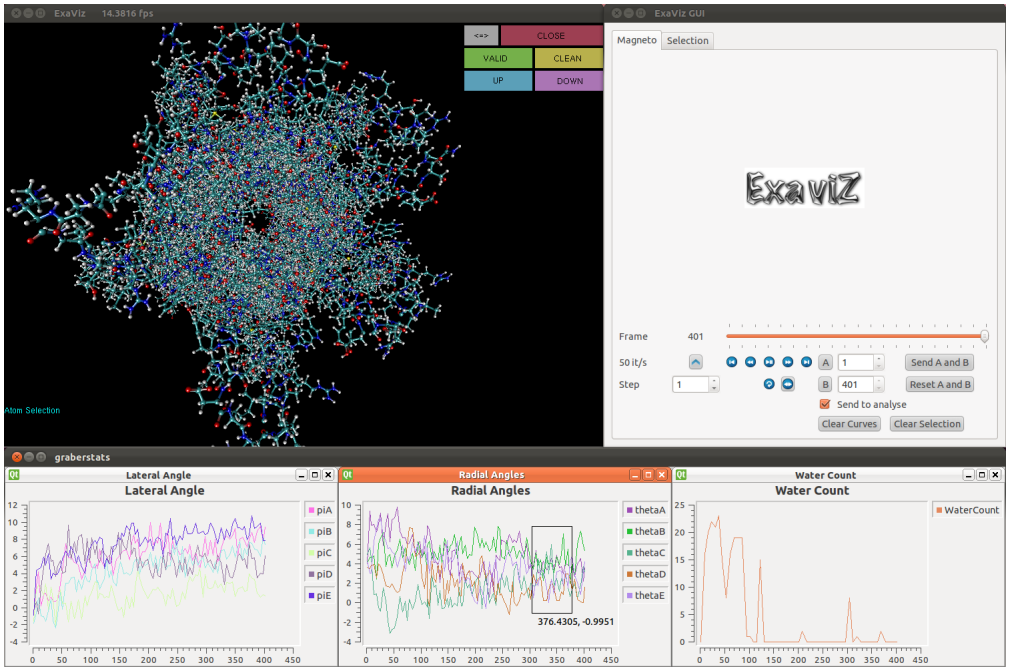

Figure 1. Screenshot of VItAMInS application

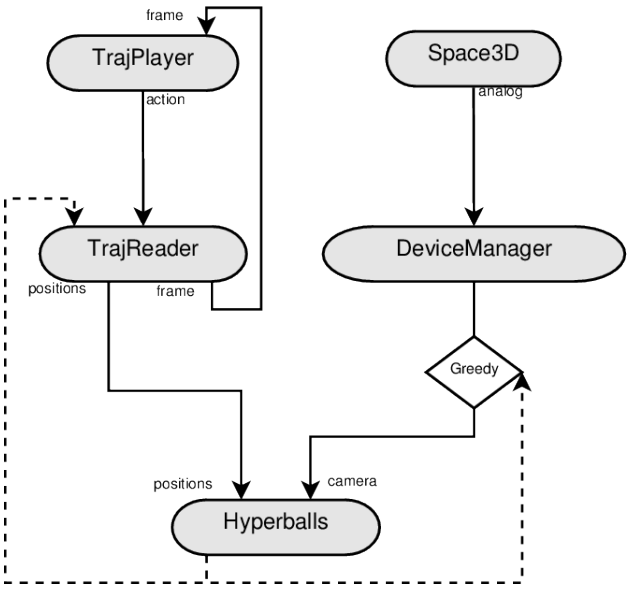

Figure 2. Example of a VItAMInS application

the last data and delivers it upon receipt of the ending signal of *Hyperballs*.

It is possible to build an in situ application from the one illustrated in Figure 2 by replacing *TrajReader* and *TrajPlayer* by *Gromacs* and an *Interface* that controls *Gromacs*. This replacement involves several modifications in the FlowVR network since *Gromacs* is a parallel module that can run an several thousands of cores which means that communications from and to this module must be done through parallel schemes. Moreover in the case of in situ analysis, if the simulation runs faster than the visualization it is usually allowed not to display all the frames. Indeed, in this way the simulation can run as fast as possible and the visualization displays all that it can. For this reason, a greedy communication will be chosen between *Gromacs* and *Hyperballs*.

The demo will illustrate the different steps to build and launch VItAMInS applications. From the basic application of post mortem analysis illustrated in Figure 2 the demo will consist in 3 parts. First, we will show the different phases to design the application from Python scripts and we will detail the way FlowVR deploys it on different platforms. In the second part, the demo will describe how to complement the *post mortem* analysis application with visual analytics modules by enriching the basic application. These parts will be based on actual trajectories studied in the French ANR project ExaViz (Figure 1).

The last part will be the description of a live analysis application to illustrate the capacities of our framework to design applications including interactions influencing the course of the simulation. We will show that the efficiency of the application is sufficient in order to achieve a rapid visualisation of all interaction results. A video for an actual system where Gromacs is deployed on around 1000 cores will be presented to illustrate the high performance computing features of VItAMInS .

References

[1] Jérémie Allard, Jean-Denis Lesage, and Bruno Raffin. Modularity for Large Virtual Reality Applications. *Presence: Teleoperators and Virtual Environments*, 19(2):142–162, April 2010.

[2] Matthieu Chavent, Antoine Vanel, Alex Tek, Bruno Levy, Sophie Robert, Bruno Raffin, and Marc Baaden. GPU-accelerated atom and dynamic bond visualization using hyperballs: A unified algorithm for balls, sticks, and hyperboloids. *Journal of Computational Chemistry*, 32(13):2924–2935, 2011.

[3] Matthieu Dreher and Bruno Raffin. A Flexible Framework for Asynchronous In Situ and In Transit Analytics for Scientific Simulations. In *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Chicago, États-Unis, May 2014. IEEE Computer Science Press.

[4] Berk Hess, Carsten Kutzner, David van der Spoel, and Erik Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*, 4(3):435–447, 2008.

[5] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. Mdanalysis: A toolkit for the analysis of molecular dynamics simulations. *J. Comput. Chem.*, 32:2319–2327, 2011.

[6] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kal, and Klaus Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

[7] Mikal Trellet, Nicolas Ferey, Marc Baaden, and Patrick Bourdot. Content-guided navigation in multimeric molecular complexes. In *International Conference on Bioimaging (BIOIMAGING 2014 2014)*, Proceedings of the International Conference on Bioimaging, pages 76–81, Angers, France, 03/03 au 06/03 2014. SciTePress.