Performance Evaluation of *in situ* Applications through Simulation using SimGrid

Context

Numerical simulations are widely used in many scientific domains to reproduce and understand complex real-life phenomenon that are difficult or impossible to study in laboratory conditions. For instance, the domain of Molecular Dynamics uses very sophisticated numerical simulations to study the behavior and interactions between molecules in a particular environment (specific temperatures etc). A numerical simulation is a complex application composed of two main phases: i) a simulation component, usually a large-scale MPI code and ii) an analysis component, a data-intensive post-processing of the simulation data to generate the final results of the application.

With modern architectures, the amount of data generated by numerical simulations led to a fundamental redesign of application workflows. The throughput and the capacity of storage subsystems have not evolved as fast as the computing power in extreme-scale supercomputers. As a result, the classical post-hoc analysis of simulation outputs (store all simulation data on disks before the analysis) became highly inefficient. *In situ* workflows have then emerged as a solution in which simulation and data analytics are intertwined through shared computing resources, thus resulting in lower latencies and better resource usage. A Data Transport Layer (DTL) is in charge of data exchange between simulation and analysis components. Only the analysis outputs have to be written on disks, avoiding the intermediate storage of massive volumes of data.

Determining the best *allocation*, i.e., how many resources to allocate to each component of an *in situ* workflow; and *mapping*, i.e., where and at which frequency to run the data analytics component, is a complex task whose performance assessment is crucial to the efficient execution of *in situ* applications. However, such a performance evaluation of different allocation and mapping strategies usually relies either on directly running them on the targeted execution environments, which can rapidly become extremely time- and resource-consuming, or on resorting to the simulation of simplified models of the components of an *in situ* application, which can lack of realism. In both cases, the validity of the performance evaluation is limited.

Goal of the project

Recently, we introduced Sim-Situ [1], a framework for the faithful performance evaluation of *in situ* processing systems. Sim-Situ builds on the popular SimGrid toolkit [2] and benefits of several key features of this versatile framework. SimGrid enables the simulation of large-scale distributed applications in a way that is accurate (via validated performance models), scalable (ability to run large scale instances on a single computer with low compute, memory, and energy footprints), and expressive (ability to simulate arbitrary platform, application, and execution scenarios). We designed Sim-Situ to reflect the typical structure of *in situ* applications with three distinct modules that respectively, (i) simulate the unmodified simulation component of the application; (ii) mimic the behavior of an underlying Data Transport Layer (DTL) that shares the data between simulation and analysis; and (iii) abstract the analysis/visualization component. Thanks to this modular design, Sim-Situ has the necessary flexibility to easily and faithfully evaluate the behavior and performance of various combinations of in situ processing system features [3].

So far, Sim-Situ is a library and is tested on a proxy-app, ExaMiniMD, that demonstrates the capabilities of Sim-Situ. Minor changes on the original code of ExaMiniMD are applied through a patch to transform the application into a in-situ framework that keeps the original simulation component and emulates the analytics one through a DTL.

The goal of this project is to further extend the expressivity and capabilities of the Sim-Situ framework. A long-term objective is to consolidate the framework in order to be a reference for the performance evaluation of *in situ* applications. In order to do so, we propose the following road-map for the research project:

- Create a Git repository with Continuous Integration: we would like to put the code of Sim-Situ in a public Git repository that would allow automatic tests of Sim-Situ with possibly different use-case applications (so far, only ExaMiniMD). This work will be the first step in the project, and allow a good understanding of Sim-Situ and its interactions with SimGrid and the underlying application.

- Test Sim-Situ with different applications: we would like to test Sim-Situ with other proxy-applications such as CabanaMD or QuickSilver. In a next effort, we expect to test the framework on more complex simulation codes (such as Gromacs [4], LAMMPS [5] etc), that are widely used in *in situ* processing. This requires to analyze the code of these applications and create a patch to add the Sim-Situ library to the application and build an *in situ* workflow.

- Code refinement: many works remain to introduce generic descriptions of simulation and analysis components. What are their common structures? How to provide a generic model in Sim-Situ?

- Externalize the application description: one of the envisionned evolution in Sim-Situ is to offer the possibility to users to describe in a dedicated file the application parameters (simulation setup, analysis component etc). This point can be closely related to the above one.

- Add new features: The evaluation of more mapping and allocation strategies needs to be investigated, especially strategies with off-node processing. Studying such strategies would be a first step in evaluating the impact of data transfers and network performance on *in situ* processing. Implementing state-of-the-art DTLs such as ADIOS [6], DataSpaces [7], or Dimes [8] is also crucial to offer flexibility to users.
  Parallel analysis are also expected to be implemented and added to the framework catalog. Only few tests have been done in this direction. The setup of the demonstration in [1] mainly relies on independent analysis workers.
  Overall, adding such features to Sim-Situ will be considered as a great asset to the framework.

Profile of the prospective student

No specific requirements are expected. A curious mind and a will to put the hands into codes would be very suitable for this topic. Knowledge on SimGrid is a plus but not mandatory. A complete tutorial [2] is available to get familiar with the toolkit. Sim-Situ is developed in C++, a good command of the langage is an advantage, especially for the aspects around code refinement. A certain taste for compilation will probably help when dealing with large-scale applications. The project can be adapted to the will and taste of the candidate.

The SimGrid framework has many developers and offers a very reactive support, with a dedicated Framateam instance. The student will be strongly encouraged to register and exchange with the community.

Regular meetings will be scheduled during the project, either on-site or remotely. We will provide access to computing resources to the student in order to ease development and tests.

Contact

Valentin Honoré valentin.honore@ensiie.fr
Associate Professor ENSIIE, Evry
Member of thze Parallel & Disitributed Systems group, Samovar

References

[1] V. Honoré, T. M. A. Do, L. Pottier, R. Ferreira da Silva, E. Deelman, and F. Suter, "SIM-SITU: A Framework for the Faithful Simulation of in situ Processing," in *eScience 2022*, Salt Lake City, United States, Oct. 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03504863

[2] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms," *JPDC*, vol. 74, no. 10, pp. 2899 – 2917, 2014.

[3] H. Childs, S. Ahern, J. Ahrens, A. Bauer, J. Bennett, E. W. Bethel, P.-T. Bremer, E. Brugger, J. Cottam, M. Dorier, S. Dutta, J. Favre, T. Fogal, S. Frey, C. Garth, B. Geveci, W. Godoy, C. Hansen, C. Harrison, B. Hentschel, J. Insley, C. Johnson, S. Klasky, A. Knoll, J. Kress, M. Larsen, J. Lofstead, K.-L. Ma, P. Malakar, J. Meredith, K. Moreland, P. Navrátil, P. O'Leary, M. Parashar, V. Pascucci, J. Patchett, T. Peterka, S. Petruzza, N. Podhorszki, D. Pugmire, M. Rasquin, S. Rizzi, D. Rogers, S. Sane, F. Sauer, R. Sisneros, H.-W. Shen, W. Usher, R. Vickery, V. Vishwanath, I. Wald, R. Wang, G. Weber, B. Whitlock, M. Wolf, H. Yu, and S. Ziegeler, "A Terminology for in situ Visualization and Analysis Systems," *IJHPCA*, vol. 34, no. 6, pp. 676–691, 2020.

[4] P. Bauer, B. Hess, and E. Lindahl, "Gromacs 2022.3 manual," Tech. Rep., Sep. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7037337

[5] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995.

[6] W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck, A. Huebl, M. Kim, J. Kress, T. Kurc, Q. Liu, J. Logan, K. Mehta, G. Ostrouchov, M. Parashar, F. Poeschel, D. Pugmire, E. Suchyta, K. Takahashi, N. Thompson, S. Tsutsumi, L. Wan, M. Wolf, K. Wu, and S. Klasky, "ADIOS 2: The Adaptable Input Output System. A framework for high-performance data management," *SoftwareX*, vol. 12, p. 100561, 2020.

[7] C. Docan, M. Parashar, and S. Klasky, "DataSpaces: an Interaction and Coordination Framework for Coupled Simulation Workflows," *Cluster Computing*, vol. 15, no. 2, pp. 163–181, 2012.

[8] F. Zhang, T. Jin, Q. Sun, M. Romanus, H. Bui, S. Klasky, and M. Parashar, "In-memory Staging and Data-Centric Task Placement for Coupled Scientific Simulation Workflows," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4147, 2017.