Dynamic Formal Verification of High Performance Runtimes and Applications

Executive summary of the PhD thesis proposal:
Recent evolutions in High Performance Computing (HPC) call to a major change of the programming models, from static large applications to adaptive and dynamic systems. Assessing the correctness of these software then constitutes a major challenge, that manual tests cannot address anymore. The goal of this work is to propose new formal verification techniques, specifically tailored to this new application domain.

Key skills required: Formal methods, runtime verification; Notions of system programming in C on Linux.

Research Unit: Inria Rennes – Bretagne atlantique

Advisors:
• Thierry Jéron (http://www.irisa.fr/prive/jeron/, team Sumo)
• Martin Quinson (http://people.irisa.fr/Martin.Quinson/, team Myriads)

Context

High performance computing (HPC) platforms have always been very challenging for the programmers in terms of size, heterogeneity and dynamicity. The design and evaluation of monolithic programs is currently becoming intractable, and the developers must rely on other approaches, more dynamic and adaptive, to deal with the HPC systems that will become available in the next decade.

Removing unnecessary synchronization constraints may be necessary because of the upcoming platforms, but it poses a major challenge to the correctness of these codes, as designing arbitrary distributed applications is notoriously difficult. The use of adapted programming models and runtime is thus very appealing to separate the applicative logic of numerical computations from concerns such as performance and correctness. A promising programming model consist in the decomposition of the numerical computations as a graph of inter-dependent computational kernels. A runtime is then used to schedule the tasks on the resources.

The goal of this thesis is to explore and propose methods to formally assess the correctness and to find bugs in modern HPC runtimes.

SimGrid\(^1\) is a scientific instrument to study the behavior of large-scale distributed computer systems. It can be used to evaluate heuristics, to prototype applications, or to assess legacy MPI applications. Performance can be used through simulation while a dynamic formal verification tool called SimGridMC is included in the framework to evaluate the correctness of protocols \([1, 2]\). The principle of this verification is to explore all the execution paths automatically and check if each execution satisfies a given property. In contrast with model checking approach that requires a model (either reconstructed automatically or manually built), the verification is performed on the real application through its controlled execution. The application’s model is thus unknown, and explored only implicitly.

Description

In order to formally assess HPC runtime, the current situation should be improved in two directions:

First, the verification process is hindered by the amount of execution paths to explore in any application, as with model-checking. This well known problem is called the state space explosion. Even small applications deployed on only a few processes easily entail billions of states to explore. SimGridMC leverages several state space reduction techniques to mitigate this problem, but more work is necessary to tackle the scales usually considered by the HPC community.

Then, making formal verification methods usable for the HPC practitioners also requires to reduce the semantic distance between the communities. This can be done in two ways. First, the properties that are

\(^1\)http://simgrid.gforge.inria.fr/
important to the HPC community should be formalized, to be then automatically enforced. Some classical properties seem interesting in HPC too, such as linearizability (the fact that a distributed service behaves as if it were atomic from an observer point of view), the absence of non-progressive cycles (inducing that the application will never enter any infinite loop) or simply the absence of crashing bugs. Along the same line, it would be interesting to discover any other properties that HPC practitioners consider important for their applications.

On the other side, it is important to better contextualize the problems diagnosed by the formal tools to make them understandable to the HPC specialists. Usually, a formal decomposition of an execution path leading to the problem is presented, which is hard to understand without a proper training. A first research point is to automatically simplify and shorten the presented execution path while removing irrelevant parts. The existing literature on root cause analysis would be inspiring to go further in that direction. Existing works in the context of theorem proving [3] propose Domain-Specific Languages used by practitioners to express the properties they are interested in, and by the tools to report the counter-examples. Adapting this approach to the context of HPC runtime verification would certainly be very interesting.

### Detailed Work Plan

After a refresh of our literature review, the first year will be devoted to the state space reduction techniques. Combining the Dynamic Partial Ordering Reduction (DPOR) and State Equality Reduction (SER) that currently exist in our tool will constitute the first theoretical challenge, because SimGridMC is an explicit verification framework with no apriori knowledge of the verified model. The SER itself could be improved with a Memory Shape Analysis approach. Other reduction techniques, based for example on live variable analysis or on the specific semantic of MPI classical functions, should then be explored.

While the contributions are expected to be evaluated on simple applications and benchmarks during the first year, the second year should be devoted to the evaluation of the contributions on real large-scale applications. The Star-PU framework\(^2\) should be used to that extend.

During the last year of this work, the student will apply his/her work on complex applications running on top of Star-PU, and write the manuscript. The ultimate goal is to assess very large modern HPC applications, such as qr_mumps [4].

### Skills required

In addition to the skills that can reasonably be expected from Master-level students, the applicant should have a good background of formal methods in general and possibly of runtime verification in particular. This could be complemented or partially compensated by a very good background in MPI programming and computationally intensive applications.

To implement the proposed techniques in SimGrid, the applicant must also be familiar with the C language under Linux, preferably with some knowledge of system or low level programming.

### References


[4] Task-based multifrontal QR solver for GPU-accelerated multicore architectures Emmanuel Agullo, Alfredo Buttari, Abdou Guermouche, Florent Lopez. [https://hal.archives-ouvertes.fr/hal-01166312/](https://hal.archives-ouvertes.fr/hal-01166312/)

\(^2\)[http://starpu.gforge.inria.fr/]