System-level State Equality Detection for the Dynamic Verification of Distributed Applications

Marion Guthmuller, Martin Quinson

System-level State Equality Detection for the Dynamic Verification of Distributed Applications

Marion Guthmuller[1] and Martin Quinson[2]

[1]PhD Student, Université de Lorraine, France (will present the poster)
[2]Assistant Professor, Université de Lorraine, France

The poster is about the joint correction and performance study of distributed applications, without any manual rewrite between models, algorithms and real implementations. Our goal is to dynamically verify unmodified legacy distributed applications written with MPI, or a similar API that allows concurrent sequential processes (CSP) to interact through message passing.

Our work aims to improve the dynamic verification tool SimGridMC, which exhaustively verifies distributed applications by executing the real program. For this, the applications can be specified using either one of the interfaces provided by the versatile SimGrid framework, or the classical MPI standard. Prior to this work, SimGridMC was stateless, only checkpointing the initial system's state. When needing to backtrack a previous state to explore another path, this initial state was first restored and then all transitions leading to the desired state were replayed. In the context of HPC applications however, this approach reveals problematic since application's computations are much more expensive than with Peer-to-Peer protocols for which SimGridMC was initially intended. It then becomes interesting to save more system states so that they can be restored without re-executing previous computation. Moreover, stateless verification only enables the verification of safety properties while the most interesting properties include vivacity. The verification of liveness properties is based on the detection of cycles during the execution of the application and thus on the detection of system's state equality. Finally, this work can also be used to exhaustively verify cyclic applications which may be infinite-time, such as cyclic protocols that are governed by periodic events (the P2P protocol Chord for example).

This poster organized as follows : we first present our motivation – the study and verification of distributed applications. As explained earlier, a preexisting version of a dynamic verification tool, SimGridMC, was developed to verify real implementations of distributed applications. This tool was however limited to the verification of one kind of property and its stateless approach wasn't suitable for the HPC applications or for some cyclic applications. The detection of system state equalities is one of major challenges for the improvement of this tool. We detail in a second part the OS-level challenges encountered when detecting the system state equality of MPI applications and propose in a third part solutions to mitigate these difficulties.

Since the byte per byte memory comparison leads to the detection of syntactic differences that are not significant, a memory introspection and a semantic comparison is necessary. We identified four major causes of these differences : (1) the memory overprovisioning implemented by many memory allocation libraries, such as the size of memory chunks is a power of two. For example, a memory area of 64 bytes will be used to serve a request of 48 bytes. Unused bytes may thus result in irrelevant differences, as they contain unspecified values ; (2) some padding bytes added by the compiler between the variables to enforce memory alignment constraints. Like with the memory overprovisioning, these bytes must not be considered in the comparison ; (3) some irrelevant differences such as PID, or file descriptors ; (4) the order in memory of malloced blocks may leads to two heaps at two different execution times syntactically different while being equal from a semantic point of view. This situation often occurs in our context because the block ordering stems from the order of malloc requests, which change when the process execution order changes, as in the dynamic verification.

For each of these OS-level issues, we propose several solutions leveraging debugging information and tools. To the best of our knowledge, this is the first heuristic that can be used on programming languages and systems without automatic garbage collection to reconstruct the needed semantic information.

We show in another part the effectiveness of our state equality mechanism through several experiments such as the dynamic verification of a liveness property on a custom MPI application and the verification of some MPICH3 integrated tests. Finally, we briefly present the future work associated to this work.