Model-Checking in a Grid Framework

1 Context

Distributed algorithms are known for being tedious to design and assess. Since they are based on the interaction of several independent components, they are hard to envision for the sequential human brains. Deadlocks, race conditions and resource starvation are unfortunately common problems of distributed computing.

These difficulties are even increased by modern computational platforms such as grids or peer-to-peer systems. Because resources are shared between users, it is hard to predict essential parameters such as bandwidth, CPU allocation or even whether the participating nodes will remain available over the time.

These uncertainties make it almost impossible to reproduce a test scenario when designing or testing software for these platforms. The use of a simulator is a common solution in the research community to help assessing the relative quality of variants of algorithms under reproducible conditions. Unfortunately, there is still ample possibility for errors being introduced when the chosen algorithm is re-implemented after simulation for use on a real platform.

The GRAS framework aims at bridging this gap. Applications developed with it can run on top of both a simulator or on real platforms without code modification, thanks to two implementations of the same interface.

One can thus build test suites alongside with distributed applications developed with this framework, and run them under different conditions within the simulator. Going beyond testing, formal methods can help to catch subtle errors that occur only in rare cases and could remain undetected otherwise.

Model checking is one of the most successful techniques to assess that a distributed algorithm exhibits the expected properties. The main idea is to exhaustively build and analyse the state space of an algorithm and check that the requirements are satisfied for all possible executions. For a distributed algorithm, all possible interleavings of the different processes will be analyzed, even for improbable differences of the relative speed of execution.

2 Internship description

The goal of the proposed internship is to enhance GRAS by allowing a developer to apply model checking techniques to algorithms developed within this framework.

Compared with model checking of arbitrary distributed applications, model checking within the GRAS framework is simplified because it can make use of the existing virtualization mechanism that makes it possible for a program to be executed within the simulation environment as well as on real platforms. GRAS programs only interact with the external world through GRAS functions, giving a full control of the GRAS execution layer over the user-code.
Moreover, when in simulation mode, a scheduler chooses the order in which the threads running the simulated processes must be awakened, depending on the availabilities of the platform. This can easily be changed to explore all possibilities.

The main missing piece needed to integrate model checking within the GRAS framework is a way to save and restore the thread state during the exploration, but several classical checkpoint mechanisms can be used for this.

3 Prerequisites

The candidate should be at ease with C programming, including POSIX threads if possible. A background on model checking and basic graph algorithms (such as finding strongly connected components) would be a plus, but is not mandatory.