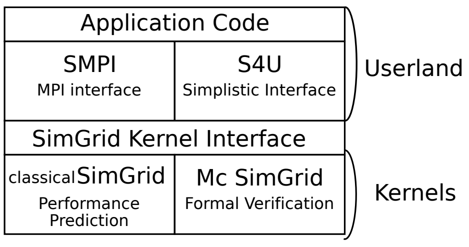
Formal Semantics of the SimGrid Simulator

June 18, 2018

This document tries to formally express the semantic of applications that can be executed in SimGrid, such as MPI applications. The long term goals is to find better reduction algorithms for MPI applications in Mc SimGrid, the model-checker embedded within the SimGrid framework.

SimGrid is a simulator of distributed applications. Several user interfaces are proposed, ranging from the classical and realistic MPI formalism, to less realistic simgrid-specific APIs that ease the expression of theoretical distributed algorithms. These user interfaces are built upon a common interface, that is implemented either on top of a performance simulator, or on top of a model-checker exploring exhaustively all possible outcomes from a given initial situation.



The distributed application is represented in SimGrid as a set of **actors**, representing processes or threads of real applications, or MPI ranks. These actors interact with each other either through message passing, or with classical synchronization objects (such as mutexes or semaphores), or through executions on CPUs and read/write operations on disks.

Even if it simulates distributed applications, SimGrid proposes a shared memory model: all actors share the same memory space. To simulate distributed settings, most of the simulated applications simply ensure that they only use variables that are local to each actor, without any program global variables. Enforcing the memory separation at application level allows the kernel to deal with shared-memory and distributed-memory primitives in the same way. It also permits to partially abstract the studied simulated infrastructure: the distributed services that are not relevant to the study can easily be abstracted as centralized components.

From the formal point of view, a major advantage of the SimGrid framework is that all user interfaces are implemented on top of a very small amount of kernel primitives. In this document, we are interested in formalizing these operations and their inter-dependencies, that will be useful for partial-order reduction methods in model-checking.

This document is organized as follows. Section 1 formally defines the programming model offered by the SimGrid kernel using TLA+. It specifies the semantic of every offered operation types through their effects on the system. Section **??** defines an event system with these operations, exploring the causality, conflict and independence relations between the defined events. Section **??** presents how the MPI semantic is implemented on top of the SimGrid kernel.

$$
\begin{array}{l}
Action \triangleq \land x' \quad = x - y \\
\land yy' \ = 123 \\
\land zzz' = zzz
\end{array}
$$

Figure 1: thu xem sao

1 System State Definition

A distributed system is a tuple $P = \langle \mathsf{Actors}, \mathsf{Network}, \mathsf{Synchronization} \rangle$ in which $\mathsf{Actors} = \{A_1, A_2, ...A_n\}$ is a set of n actors. Actors do not have a global shared memory nor a global clock. The execution of an actor A_i consists of an alternate sequence of local states and actions $s_{i,0} \xrightarrow{a_0} s_{i,1} \xrightarrow{a_1} s_{i,2} ... \xrightarrow{a_{n-1}} s_{i,n}$ (firing a_i from local state $s_{i,j}$, the state of actor A_i changes from $s_{i,j}$ to $s_{i,j+1}$). All the actions in one actor are totally ordered by the causal relation. The subsystem $\mathsf{Network}$ provides facilities for the Actors able exchange messages with each other while subsystem $\mathsf{Synchronization}$ composed of several mutexes to synchronize actors when they access shared resources.