A modular and uniform programming environment for high-level distributed computing

Mots clés :

- Directeur de thèse : Marc Shapiro
- Co directeur de thèse : Ahmed BOUAJJANI
- Co-encadrant(s) :
- Unité de recherche : Laboratoire d'informatique de Paris 6
- Ecole doctorale : École Doctorale Informatique, Télécommunications, Électronique de Paris
- Domaine scientifique principal: Sciences de l'information et de la communication

Résumé du projet de recherche (Langue 1)

The goal of the PhD is to identify the abstractions needed for writing distributed applications or distributed systems and to ease distributed programming for non experts. Distributed programming is by nature concurrent, error-prone and subjected to variable latency which make computations highly non-deterministic. Furthermore, applications have conflicting requirements: correctness, i.e. controlling what the system does, versus application performance, which includes availability, responsiveness or throughput. There is no single right solution to this trade-off; it depends on application requirements, environment, workload, the available resources, etc. One common approach to address high-level distributed programming is to restrict expressiveness, as for instance with TensorFlow or MapReduce. The closest to a general-purpose programming environment is Orleans: however, part of its runtime (e.g. deployment, consistency, security) is provided by a separate subsystem, and not programmable with the same first-class abstractions. This is ripe for a general-purpose approach for both application developers who should not be bothered with low-level details, and system developers, who require fine control at runtime. It should be possible for both to create, protect, ensure correctness and use powerful abstractions, making first-class what is built-in in the above systems. An abstraction would be able to compose basic primitives that provide access to the full power of distribution. We aim to provide a uniform and modular programming environment for this purpose. This will increase the reliability of such programs, by helping to ensure correctness at all levels of the environment. The environment should include: (1) a programming model, to be able to reason precisely about the behaviors of systems; (2) a language to be able to write applications and systems; and (3) a runtime system to assist in running and deploying code and data on a distributed infrastructure. In the first year we will study state-of-the-art and derive requirements. The second year will be devoted to implementing the language and a runtime programmable as a first-class abstractions. Moreover, this will generate useful feedback. Finally, we plan to run a large experimental evaluation, to assess success and provide a reality check, for instance by implementing a geo-distributed database.