A distributed runtime for composable distributed computing

Mots clés :

- Directeur de thèse : Mesaac Makpangou
- 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 et technologies de l'information et de la communication

Résumé du projet de recherche (Langue 1)

Distributed computing has become crucial across many areas of computing. Programming highly-available distributed applications is no longer reserved to expert programmers: consider Internet of Things, High Performance Computing, or internet gaming. However, programming distributed systems remains difficult and error-prone, exposing users and critical infrastructure to bugs. Furthermore, applications have to scale from a single node to geo-distributed infrastructures. Moreover, distributed systems need to deal with failures, non-determinism, asynchrony and concurrency. This leads to trade-offs (e.g. CAP, FLP) without a one-size-fits-all solution. Hence, a flexible, dynamically evolving and customizable framework is required. This PhD research aims to alleviate this complexity, by proposing a run-time platform for composing distributed abstractions into a working system. The platform will include a library of base abstractions, including programming abstractions, such as processes and actors; communication abstractions, such as events and channels; and extra-functional abstractions, such as placement and load-balancing. This enables the developer to focus on the business requirements, and delegate the complexity of distribution to appropriate platform components. The developer can reuse or extend the base abstractions, and compose them into a functional distributed application. Many of these abstractions have already been identified in the literature, but in existing system implementations, they remain entangled and non-modular. A major scientific challenge will be to identify and implement elementary, composable components. For instance, there is a split between event-based vs. sharing-based systems, which offer very different programming models that do not compose well. We believe that they can be reunified thanks to a careful choice of consistency model: under causal consistency, any observable state includes all the events that led to that state, and any event enables the receiver to observe the states that led to that event. Nonetheless, the consistency protocol itself should constitute a separate and replaceable component. The proposed approach will ease the development of high-performance distributed applications, by making it possible to build a complex distributed application by composing simple components, ultimately delivering complex distributed systems concepts to mainstream application programmers.