Language Extension Proposals for Cloud-Based Computing

Adrian Francalanza
adrian.francalanza@um.edu.mt, and Tyron Zerafa
tzer0001@um.edu.mt

University of Malta

1 Synopsis

Cloud computing can be described as the homogenisation of resources distributed across computing nodes, so as to facilitate their sharing by a number of programs. In some sense, the use of virtual machines in languages such as Java and Erlang, are a first step towards to this idea of cloud computing, by providing a common layer of abstraction over nodes with different characteristic. This has fostered various forms of distributed computing mechanisms such as web services based on remote procedure calls (RPCs) and code on demand (COD) applets executing in sandboxed environment.

Erlang is an actor-based programming language that lends itself well to the construction of distributed programming through a combination of language features such as message-passing and error-handling mechanisms. It also offers mechanisms for dynamically spawning processes (actors, to be more precise) on remote nodes, for reasons ranging from load balancing to the maximisation of computation proximity vis-a-vis the resources that it uses. Although the mechanism work well, it relies on a rather strong assumption when programming for the cloud, namely that the source-code at every node is homogeneous.

The first aim of our study is to create a layer of abstraction that automates the necessary migration of source-code so as to allow seamless spawning of processes across nodes "on the cloud". The challenge here is to migrate the least amount of code, at the least amount of cost/effort, so as to allow the remote computation to be carried out. The solutions we explore range from those that rely on code dependency static analyses to "lazy" dynamic methods that migrate source code only when needed by the host node. There are also issues relating to source-name clashes and versioning.

The second aim of the study is to enhance the security of resources advertised on the cloud, by delineating their expected use. Our approach will rely on the mechanism of having a policy file per node, describing the restrictions imposed on resource usage. We plan to explore feasibility of code migration mechanisms that take into consideration these policy restrictions imposed by the receiving nodes. We shall also explore various ways how to enforce these policies.

The third aim of the study is to analyse mechanisms for tolerating network errors such as node disconnections; this is particularly relevant to distributed computations spanning over more that two nodes. Again, we plan to use the policy-file approach to automate decisions that need to be taken to seamlessly carry out distributed computation in the eventuality of such failures; the policy files may also be used to specify redundancy mechanisms that should, in turn, enable better tolerance to such faults.