Benchmarking Communication in Actor- and Agent-Based Languages

Rafael C. Cardoso[1], Jomi F. Hübner[2], and Rafael H. Bordini[1]

[1] FACIN–PUCRS
Porto Alegre - RS, Brazil
rafael.caue@acad.pucrs.br
r.bordini@pucrs.br

[2] DAS–UFSC
Florianópolis - SC, Brazil
jomi@das.ufsc.br

Abstract. This paper presents the results of communication benchmarks comparing an agent-oriented programming language and two actor-oriented programming languages. It is based on an existing benchmark for programming languages and two variations on that benchmark. We selected Erlang and Akka (using the Scala interface) to represent actor languages, and Jason as the agent language representative. We also present those three scenarios and the respective results in regards to time, core usage, and memory. Even though BDI engines typically used for agent languages provide sophisticated programming abstractions that require significant platform overhead to facilitate the development of complex agents, our initial results show that Jason has reasonable performance for this type of benchmark, where actor-based languages were expected to do significantly better than agent languages.

Keywords: benchmarking, agents, actors, Jason, Erlang, Akka, Scala.

1 Introduction

Jason is one of the best-known platforms for the development of multi-agent systems based on agent-oriented programming. Various authors have included Jason in their comparisons and analyses of agent programming languages. For example, Jason was included in a qualitative comparison of features available in Erlang, Jason, and Java [21]; in a universal criteria catalog for agent development artifacts [11]; in a quantitative analysis of 2APL, GOAL, and Jason regarding their similarity and the time it takes for them to reach specific states [7]; a performance evaluation of Jason when used for distributed crowd simulations [17]; an approach to query caching and a performance analysis of its usage in Jason, 2APL, and GOAL [3]; and finally an implementation of Jason in Erlang and a benchmark for evaluating its performance [16]. In those cases where performance was considered, Jason typically showed excellent results.

M. Cossentino et al. (Eds.): EMAS 2013, LNAI 8245, pp. 58–77, 2013.

However, there is no quantitative analysis — to the best of our knowledge — of how well agent languages can do compared actor languages. Because the actor approach is by design lighter than agents and because actor languages have been improved over a much longer period than modern agent programming languages, comparing performance on traditional programming language benchmarks is a much harder challenge for Jason than those it previously faced, and this is precisely what we do in this paper.

Our motivation for this recent line of work came from the idea of doing some benchmarking experiments in order to investigate whether some variations of usual benchmarking scenarios, taking into consideration features of agent programming, would allow us to conclude whether certain scenarios could be more appropriate for actors rather than agents and vice-versa, both in terms of naturalness of the paradigm for developing the applications and in terms of actual performance for the natural solutions in both paradigms. Even if it turns out that we cannot immediately find a scenario that is intrinsically more appropriate for agents, we could still use the outcome of our work to point out some flaws or deficiencies in current agent-based languages, and learn something from the longer experience with actor-based languages, thus making it possible to improve performance for agent-based languages in the future.

We started by taking an Erlang program for a token passing problem available in the Computer Language Benchmarks Game website (`http://shootout.alioth.debian.org/`) and we wrote a Jason and Akka version for it. We then changed that benchmark to a different scenario where the only difference is that a number of tokens were being passed simultaneously, and all three programs were changed accordingly. While Jason had the worst performance in regards to elapsed time, it matched closely the performance of Akka, at least for our current experiments. Erlang showed the best performance in all cases, which was expected as it is known to have an efficient virtual machine and used in industrial applications. Finally, for the third scenario, we added a notion of token types in order to assess the reactivity of each language[1].

The results reported in this paper were obtained from runs on a dedicated computer with six physical cores (no hyperthreading). We also show the results for the same scenarios but limiting the number of cores to three, with the purpose of analysing the difference that it makes to run the same experiment on increasing numbers of cores. The experiments presented in this paper are not supposed to stress-test the languages but rather to compare them in normal day-to-day usage measuring the performance, scalability, and reactivity of the communication aspect of the languages. Because this comparison cannot be done directly, as the actor model is by design lighter than the agent model and each language has a different runtime environment, we make use of scale factors to compare the languages.

[1] The code for all scenarios/languages used in this paper is available at `https://github.com/rafaelcaue/Actor-Agent-based-benchmark-for-communication`.