Universiteit van Amsterdam

Programmeertalen

Schrijfopdracht

L. Thomas van Binsbergen

Case study: Akka

Deadline: Maandag, Week 6

Practical information

Before answering questions: read through the whole assignment and discuss the resources (below) within your group.

Make sure you include your names, student numbers and group number in your submission. This assignment is executed in a group of three. Only submit one solution per group. Groups are self-registered on Canvas. You can write your answers in any medium you like (text editor, markdown editor, LaTeX/Overleaf, Word, etc.) as long as you **submit a pdf document**. If in your answers you wish to include code fragments, then working in LaTeX, markdown, or plain text are recommended. Write specific and concise answers and include references when citing or quoting outside material. The questions with point indicators sum up to 80 marks out of a maximum of a 100. At least some answers to 'bonus points' questions are therefore required to score above an 8.0 for this assignment.

Introduction

The Akka toolkit provides libraries in both Scala and Java for *actor-oriented* programming, a style of concurrent programming based on the 'actor' concept. The Akka libraries (whether in Scala or Java) effectively define a language. Languages provided by libraries in this way are referred to as *embedded languages*. In this case, Akka is embedded in Scala or Java, depending on which version of the library you use (Scala or Java is then referred to as the *host language*). Embedded languages 'borrow' the syntax and semantics of their host language, but use these to provide additional concepts (such as actors) and operators (such as message-passing functions) for programmers to work with.

The purpose of this assignment is to investigate actor-oriented programming in the embedded language Akka. To do so, you will use the following resource. As you will see, the documentation applies to both the Java and Scala version of Akka. You therefore do not have to choose one version to make this assignment.

- The documentation of the current Akka version: `https://doc.akka.io/docs/akka/current/`

Akka's actor model

The actor model of Akka addresses two major challenges associated with concurrent (multi-threaded) programming. The first is that in a context in which multiple threads execute concurrently, the same code block (e.g. in a method/-procedure) can be executed simultaneously by multiple threads. This is an issue if these simultaneous executions of a code block affect the same memory locations, resulting in corrupted state and unexpected results. The concept of a 'lock' – found for example in object-oriented programming languages with concurrency – addresses this issue by 'locking' code blocks for other threads once a first thread has started executing the block. However, locking code blocks can have significant performance penalties, and deciding where to place the locks without causing performance issues can be a real challenge.

Q1 (10 points) In your own words, explain how message-passing in Akka avoids the problem of corrupted state, without having to rely on the programmer for avoiding the problem (e.g. by introducing locks to the code). In your answer, make explicit what is meant by the state of an actor and how messages are processed by an actor.

Q2 (10 points) According to Akka's documentation, its message-delivery system guarantees two properties about the delivery of messages. In your own words, explain these properties and their relevance.

Q3 (15 points) In your own words, why does Akka not guarantee the delivery of messages? What are the arguments made to this design decision by the documentation? Mention at least three reasons.

The second major issue addressed by Akka's actor model is that of *failures* (unexpected problems caused by, for example, a bug in the code or network/disc error) and *exceptions* (foreseeable, yet exceptional situations).

Q4 (20 points) In your own words, what mechanism(s) does Akka provide to handle failures? In your answer, mention the hierarchy formed by Akka actors and different possible responses to an observed failure. Be specific.

Q5 (10 points) A significant difference between a message being processed and a method/function/procedure being called, is that processing a message does not return a value. What does the Akka documentation suggest a programmer do instead? In your answer, refer to at least one of the 'interaction patterns' described in the documentation.

Q6 (bonus points) Explain an additional interaction pattern and motivate it through an example of your own choosing

Embedded nature of Akka

The following questions relate to the fact that Akka is an embedded language (as explained in the introduction).

Q7 (10 points) In either Java or Scala, is Akka a compiled or interpreted language? Explain how you concluded this. If interpreted, in what programming language is the interpreter written? If compiled, what is the target language/instruction set of the compiler?

Q8 (5 points) In either Java or Scala, is Akka statically or dynamically typed? How did you conclude this?

Q9 (bonus points) Describe an application for which you think Akka's actor-oriented programming is particularly suitable. Argue whether for this application you would prefer to use the Java's Akka, Scala's Akka, or Erlang.

Q10 (bonus points) How can Akka programs be debugged? Is this simpler or more difficult than in other languages you are familiar with? How so?

Additional bonus questions

Q11 (bonus points) The following page contains links to several additional Akka modules that extend the functionality of Akka even further. For one of these modules: explain what it is for by mentioning the kinds of applications that can benefit from using this module. Pick an example application of your own choosing.
`https://doc.akka.io/docs/akka/current/common/other-modules.html`

Q12 (bonus points) Based on your observations, what are the main similarities and differences between Akka and Erlang? You can refer to syntactic, semantic, or pragmatic aspects of the languages. Do the languages differ in terms of the concepts associated with the programming paradigms discussed in this course?

Q13 (bonus points) Write a concurrency abstraction using Akka actors that implements remote procedure calls. The goal of this abstraction is to make it possible to just execute a function/method that does the communication for us and waits for a response which it returns as the result of the call. You can use either Scala or Java or pseudo-code resembling Scala or Java.