Development of concurrent and distributed programs with the Actor model and Akka

ugr | Universidad de Granada

César Aguilera Padilla

Thematic CERN School of Computing
Split, 27th of May 2016

Overview

- Motivation
- What is the Actor model?
- When can we use the actor model?
- What are its advantages and drawbacks?
- What is Akka?
- Conclusions

Motivation

- Writing concurrent and parallel programs is tough
- Programming with threads, locks, atomic transactions, etc… is error prone and lead us to write code difficult to read, test and maintain

Motivation II

- Outcome 1: panic to code multi-threaded programs
- Outcome 2: single threaded applications relying on external services (Databases, file systems…) in order to handle concurrency and/or asynchronous operations for them

Motivation III

- Is there an alternative to deal with threads, locks, semaphores and race conditions if we want to build concurrent/parallel applications?

Actor model!

Actor model

- Mathematical model of concurrent computation proposed by Carl Hewitt in 1973
- It faces concurrent problems from a quite different perspective

Actor model II

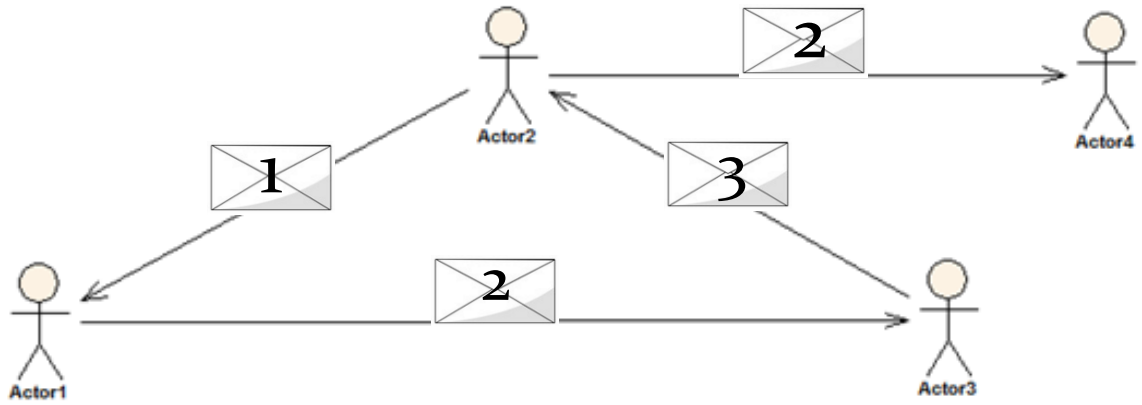
- Actors→ primitives for concurrency/parallelism
- Actors → Entities having a message queue and associated behavior → And isolated state!!
- Actors→ Can exchange messages between each-other
- Actors → When an actor receives a message it can:
  - Send a finite number of messages to other actors
  - Create a finite number of new actors
  - Modify its interval behavior on receiving messages

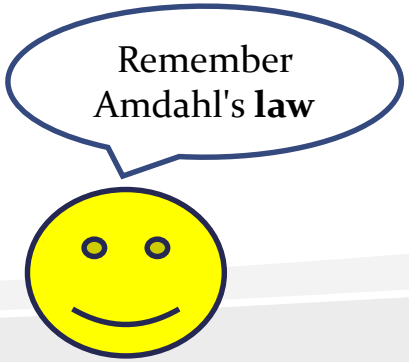
Actor model III

- Messages between actors are always sent asynchronously
- No requirement on order of message arrival
- Queuing and dequeuing of messages in an actor mailbox are atomic operations, no race conditions anymore

¿When can we use the actor model?

- When…
  - The problem to be solved can be decomposed into a set of independent tasks
  - The problem to be solved can be decomposed into a series of tasks linked by a clear flow
- In short words… when the problem can be parallelized





Advantages and Drawbacks

- Extends the benefits of object-oriented programming by splitting control flow and business logic
- Allows to decompose a system into interactive, autonomous and independent components that work asynchronously

- Sometimes creating actors may dramatically affect the system's responsiveness
- The decision of where to store and run the new actors requires to archive a series of records, so if it is not done well it can lead to performance penalties in highly distributed systems

What is Akka?

- Akka is an open-source framework for Java and Scala that simplifies building concurrent and distributed systems on the Java Virtual Machine
- Aimed to solve concurrent/parallel problems by using the actor model

Akka main features

- Provides simple and high-level abstractions for concurrency and parallelism
- Provides a high performant programming model based on events, asynchronous and non-blocking
- Allows to build hierarchical networks of actors, ideal for building fault-tolerant applications
- It allows to deploy networks of actors in different Java Virtual Machines and easily interconnect with each other
- It is purely designed to operate in distributed environments

Conclusions

- The actor model is (yet) another approach to develop concurrent and distributed applications
- Allows to easily model concurrent and distributed systems through high-level constructions → Actors
- Allows to create software systems based on independent, interactive and autonomous components that interoperate asynchronously
- Has several implementations, most famous is Akka but there is also an implementation for C++ ☺

References and interesting links

- Carl Hewitt, Peter Bishop, Richard Steiger. “A Universal Modular Actor Formalism for Artificial Intelligence”. IJCAI, 1973.
- Akka.io, “Untyped Actors”, http://doc.akka.io/docs/akka/2.3.7/java/untyped-actors.html.
- Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters, Google, 2004.
- write2munish on GitHub. “Akka-Essentials examples”. https://github.com/write2munish/Akka-Essentials/tree/master/FirstAkkaApplication/src/main
- Benjamin Erb, “Concurrent Programming for Scalable Web Architectures”, Institute of Distributed Systems, Ulm University, 2012.
- Diego Castorina. Concurrency and Fault Tolerance Made Easy: An Intro to Akka. http://www.toptal.com/scala/concurrency-and-fault-tolerance-made-easy-an-intro-to-akka.
- Wyatt, Derek. “Akka Concurrency”, 1st ed. Artima Press, 2013.
- Gupta, Munish K. “Akka essentials”, 1st ed. Packt Publishing, 2012.
- Daniel Westheide. “The Neophyte's Guide to Scala Part 14: The Actor Approach to Concurrency”, http://danielwestheide.com/blog/2013/02/27/the-neophytes-guide-to-scala-part-14-the-actor-approach-to-concurrency.html.