PL@NES - Reading Club

Actors à la Akka

Christophe.VanGinneken@cs.kuleuven.be
THE FOLLOWING **PRESENTATION** IS NOT ABOUT A **SCIENTIFIC** PAPER. IT INTRODUCES AN **INDUSTRY**-GRADE TECHNOLOGY THAT IS REPRESENTATIVE FOR THE LEVEL OF ADOPTION OF **THE ACTOR MODEL**. OR MAYBE NOT…

**RECOMMENDED**

MIGHT CONTAIN MILD HUMOR. MIGHT PROVOKE INTELLECTUAL DISCUSSIONS AND CAUSE FRESH IDEAS TO EMERGE.
STALLONE

RAMBO I

“FIRST BLOOD”

MARIO KASSAR et ANDREW VAJNA Présentent
SYLVESTER STALLONE

“RAMBO I”

RICHARD CRENNA BRYAN DENNEHY

Un film de TED KOTCHEF. Musique de JERRY GOLDSMITH

Directeur de la photographie ANDREW LASZLO. Producteurs exécutifs MARIO KAISAR
et ANDREW VAJNA. Coproducteur exécutif HERB NANNAS. Produit par BUZZ FEITSHANS
What are Actors?
What are Actors?

Actor (disambiguation)

From Wikipedia, the free encyclopedia

An **actor** is a person who plays a role in theater, cinema or television.

**Actor** can also refer to:

- Actants, also called actors, in actor-network theory (a general theory of sociological behaviour), the one who performs the act
- in **Interactions of Actor Theory**, excitations in any medium able to produce action, a theory of cybernetics
- in computing:
  - Actor (UML), requirements analysis and UML
  - Actor model, in concurrency, refers to a model of concurrent computation
  - Actor (programming) (OOP), integrated development environment (IDE) for the Windows operating system
- **Actor (law)**
- **Actor (mythology)**, in Greek mythology, refers to a number of characters, including the father of Menoetius and Astyoche
- **Actor (policy debate)**, the entity that enacts a certain policy action
- **Actor (album)**, a 2009 album by St. Vincent
The Actor Model

source: https://en.wikipedia.org/wiki/Actor_model
The actor model in computer science is a mathematical model of concurrent computation that treats "actors" as the universal primitives of concurrent computation: in response to a message that it receives, an actor can make local decisions, create more actors, send more messages, and determine how to respond to the next message received.
According to Carl Hewitt, unlike previous models of computation, the Actor model was inspired by physics, including general relativity and quantum mechanics. It was also influenced by the programming languages Lisp, Simula and early versions of Smalltalk, as well as capability-based systems and packet switching. Its development was "motivated by the prospect of highly parallel computing machines consisting of dozens, hundreds or even thousands of independent microprocessors, each with its own local memory and communications processor, communicating via a high-performance communications network."[2] Since that time, the advent of massive concurrency through multi-core computer architectures has revived interest in the Actor model.
The Actor Model

Fundamental concepts

The Actor model adopts the philosophy that everything is an actor. This is similar to the everything is an object philosophy used by some object-oriented programming languages, but differs in that object-oriented software is typically executed sequentially, while the Actor model is inherently concurrent.

An actor is a computational entity that, in response to a message it receives, can concurrently:

- send a finite number of messages to other actors;
- create a finite number of new actors;
- designate the behavior to be used for the next message it receives.

There is no assumed sequence to the above actions and they could be carried out in parallel.
The Actor Model

Applications

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. (December 2006)

The Actors model can be used as a framework for modelling, understanding, and reasoning about, a wide range of concurrent systems. For example:

- **Electronic mail** (e-mail) can be modeled as an Actor system. Accounts are modeled as Actors and email addresses as Actor addresses.

- **Web Services** can be modeled with SOAP endpoints modeled as Actor addresses.

- **Objects with locks** (e.g., as in Java and C#) can be modeled as a Serializer, provided that their implementations are such that messages can continually arrive (perhaps by being stored in an internal queue). A serializer is an important kind of Actor defined by the property that it is continually available to the arrival of new messages; every message sent to a serializer is guaranteed to arrive.

source: https://en.wikipedia.org/wiki/Actor_model
The Actor Model

The Actor Model

Wireless Sensor Network?

OOP vs Actor Model

Sequential Control

Class
  data
  methods

input

Actor
  data
  params/ports

Streams of Data

“output”

return

call

OOP vs Actor Model

OO Interface
Procedures to be invoked in sequence.

Actor Interface
"Give me text, I’ll give (you) Speech"

OOP vs Actor Model

OOP vs Actor Model

When the actor model was first proposed, the development of distributed networks was in its infancy. The conceptual model of actors is easy to understand as it allows state to be directly expressed. Also the only side effects of an actor are to send communications and to set a new behaviour. The simplicity of this model suggests that it would make programming for a distributed system simpler, but there proved to be difficulties associated with its implementation.

- No notion of inheritance/hierarchy
- Changing behaviour (storage,...)
- Dynamic behaviour versus static languages
- Asynchronous messaging versus algorithms

source: http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol2/pjm2/
STALLONE

They sent him on a mission and set him up to fail. But they made one mistake.

They forgot they were dealing with Rambo.

RAMBO
FIRST BLOOD PART II
What is Akka?
Scalable real-time transaction processing

We believe that writing correct concurrent, fault-tolerant and scalable applications is too hard. Most of the time, it's because we are using the wrong tools and the wrong level of abstraction. Akka is here to change that. Using the Actor Model, we raise the abstraction level and provide a better platform to build scalable, resilient and responsive applications—see the Reactive Manifesto for more details.

For fault-tolerance we adopt the "let it crash" model which the telecom industry has used with success. "Let it crash" means to build applications that self-heal and systems that never stop. Actors also provide the abstraction needed to build fault-tolerant applications.

Akka is Open Source and available under the Apache 2 License.
package sample.hello;

import akka.actor.Props;
import akka.actor.UntypedActor;
import akka.actor.ActorRef;

public class HelloWorld extends UntypedActor {
  @Override
  public void preStart() {
    // create the greeter actor
    final ActorRef greeter = getContext().actorOf(Props.create(Greeter.class), "greeter");
    // tell it to perform the greeting
    greeter.tell(Greeter.Msg.GREET, getSelf());
  }

  @Override
  public void onReceive(Object msg) {
    if (msg == Greeter.Msg.DONE) {
      // when the greeter is done, stop this actor and with it the app
      getContext().stop(getSelf());
    } else
      unhandled(msg);
  }
}

package sample.hello;

import akka.actor.UntypedActor;

public class Greeter extends UntypedActor {
  public static enum Msg {
    GREET, DONE;
  }

  @Override
  public void onReceive(Object msg) {
    if (msg == Msg.GREET) {
      System.out.println("Hello World!");
      getSender().tell(Msg.DONE, getSelf());
    } else
      unhandled(msg);
  }
}

package sample.hello;

public class Main {
  public static void main(String[] args) {
    akka.Main.main(new String[] { HelloWorld.class.getName() });
  }
}
Actors à la Akka

purely reactive component
(act on receive)
create

Actors à la Akka

system.actorOf(props, "Foo")

addressing/selection

context.actorSelection("/Bar/E")
Actors à la Akka

- **ActorPath**
  - Empty path

- **Actor incarnation**
  - Belongs to a path
  - Has a UID
  - Has a Mailbox

- **Actor instance**
  -Resume
  - Restart
  - Stop or context.stop() or PositionPill

- **New instance**
  - New instance replaces old
  - postRestart() called on new instance

- **ActorRef**
  - Represents the incarnation
  - Hides the instance
  - Has a path
  - Has a UID

- **ActorSelection**
  - Represents a path (or multiple with wildcards)
  - Allows resolving the underlying ActorRef by sending an Identify message

- **ActorIdentity**
  - Identify
context.actorSelection("/Foo/A").send(msg)

Message Delivery Reliability

1. at-most-once delivery
DEAL WITH IT.
A

B

C

P

context.actorSelection("/Foo/A").send(msg)

Message Delivery Reliability

1. **at-most-once** delivery
2. message ordering per sender-receiver pair

Actors à la Akka
Actors à la Akka

```java
class HotSwapActor extends UntypedActor {
  Procedure<Object> angry = new Procedure<Object>() {
    @Override
    public void apply(Object message) {
      if (message.equals("bar")) {
        getSender().tell("I am already angry?", getSelf());
      } else if (message.equals("foo")) {
        getContext().become(happy);
      }
    }
  };

  Procedure<Object> happy = new Procedure<Object>() {
    @Override
    public void apply(Object message) {
      if (message.equals("bar")) {
        getSender().tell("I am already happy :-)", getSelf());
      } else if (message.equals("foo")) {
        getContext().become(angry);
      }
    }
  };

  public void onReceive(Object message) {
    if (message.equals("bar")) {
      getContext().become(angry);
    } else if (message.equals("foo")) {
      getContext().become(happy);
    } else {
      unhandled(message);
    }
  }
}
```
create +remoting

Actors à la Akka

context.actorSelection("akka.tcp://system@hostY:1234/Bar/E")

system.actorOf(props, "Foo")

hostX

hostY

system.actorOf(props, "Foo")

addressing/selection

A

B

C

D

E

/Foo/A

/Foo

/Foo

/Foo

Bar

Bar

Bar

Bar
Actors à la Akka

The Phi Accrual Failure Detector
http://ddg.jaist.ac.jp/pub/HDY+04.pdf

Routers

Remote Events
Actors à la Akka

Ring-structured Cluster
à la Dynamo, Riak

Gossip Protocol
for membership, leader determination, configuration

Vector Clocks

Leaders are not elected
Actors à la Akka

- Create, send, become
- Parents handle failures
- Purely reactive components
- Remoting with basic guarantees
- Clustering
Actors à la Akka

http://2013.flatmap.no/klang.html
PL@NES - Reading Club
Actors à la Akka

Christophe.VanGinneken@cs.kuleuven.be

http://www.slideshare.net/christophevg/actors-la-akka