Distributed Processes and Actors

One Paradigm to Rule them All ?

by Nils Schnabl/HAW Hamburg/TI

There are 4 process communication paradigms :

- Shared Memory with Locks
- Software Transactional Memory (STM)
- Futures, Promises and Similar
- Message Passing

The actors are message passing entities

As the fundamental unit of computation it has to embody :

0	• processing
1	• storage
2	• communication

One actor is no actor, they come in systems

„Carl Hewitt“

An actor can :

I	• create more actors
II	• send messages to others
III	• change his behavior

Asynchronous Message Passing

- each actor has an mail address
- messages can arrive in any order
- no intermediaries
- ordering guarantees must be implemented
- actors don‘t block ressources

States & Actor model

- actors change state over time
- each actor has its own state due to time
- there is no global state
- if I want to have a system state, I have to collect all states from each actor

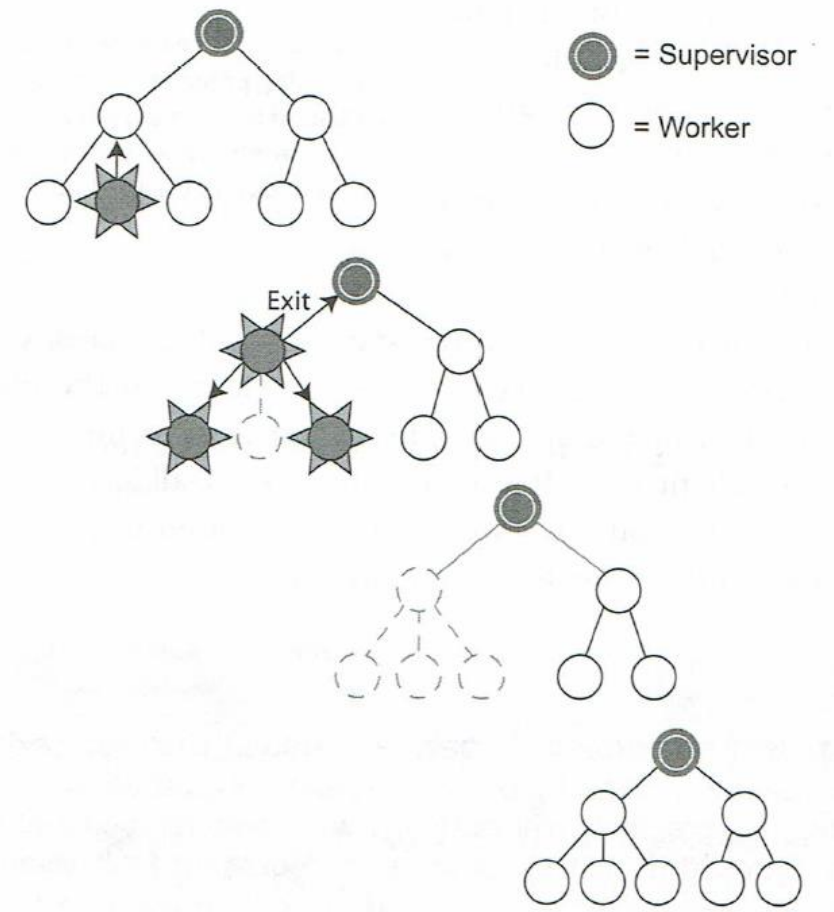
History of the Actor model

- Hewitt, Bishop and Steiger's (1973) publication were inspired by physics
- Gul Agha's (1985) dissertation
- this resulted in the full development of actor model theory
- in the 80's at Ericsson's Laboratory in Stockholm it became clear that no language had a suitable concurrency model
- Erlang took shape around 1988 by Joe Armstrong
- releasing Erlang as open source happened in 1998

Erlang/OTP

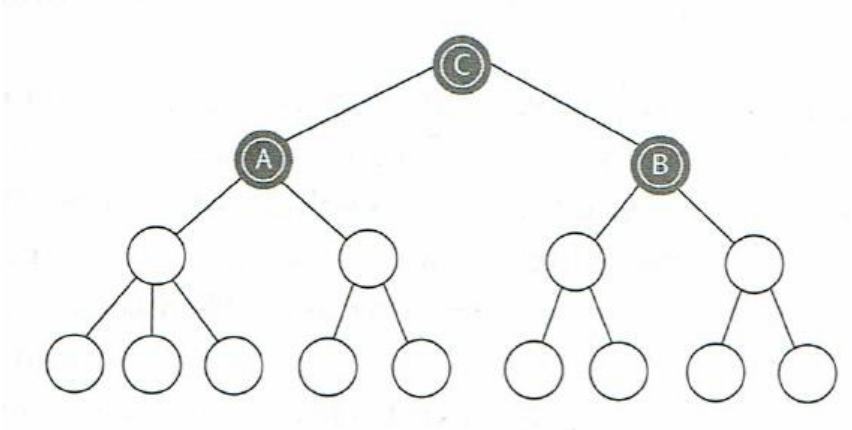
1	• Erlang is a functional programming language
2	• „Standard Library“ is OTP (Open Telecom Platform)
3	• a process in Erlang is an actor
4	• Erlang focuses failure isolation
5	• location transperancy

Erlang/OTP - Supervision



- error propagation for exit signals
- worker processes are linked
- supervisor restarts the group
- the other group of processes under the same supervisor isn‘t affected

Erlang/OTP – Layering processes



- a layered system of supervisors and workers
- if supervisor A dies or gives up, any still-living processes under it are killed and supervisor C is informed
- the whole left-side process tree can be restarted
- supervisor B isn‘t affected unless C decides to shut everything down

OTP goals

Productivity	• high level building blocks
Stability	• solid praxis proven components
Supervision	• the application structure provided by OTP makes it simple to supervise and control the running system
Upgradability	• provides patterns for handling systematic code upgrades
Reliable code base	• OTP is rock solid and has been thoroughly battle tested

Akka

- is an event-driven middleware framework
- is for applications in Java and Scala
- Erlang-like actor implementation
- Strong focus on configurability
- part of the Scala Standard Library

Erlang & Akka

- Actor communication can be strongly type-safe
- Erlang works with dynamic typing
- Akka accepts all message-types, because all messages are encapsulated in objects

CAF – C++ Actor Framework

- has possibilities to make all communication type-safe
- actors in CAF are lightweight, consisting of only a few hundred bytes
- is taking care of the low-level side of things
- Message passing is network transparent, actors can talk to each other, no matter where they've been spawned

Pony

- is an object-oriented, **actor-model**, capabilities-secure programming language
- Correctness. Incorrectness is simply not allowed
- Performance. Runtime speed is more important than everything except correctness
- if compiling, it will also working
- you can’t build a runtime error

References

- C. Hewitt, P. Bishop, R. Steiger, A Universal Modular ACTOR Formalism for Artificial Intelligence, in: Proceedings of the 3rd IJCAI, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1973, pp. 235–245.
- G. Agha, Actors: A Model of Concurrent Computation In Distributed Systems, Tech. Rep. 844, MIT, Cambridge, MA, USA, 1986.
- M. Logan, E. Merritt, R. Carlsson, Erlang and OTP in Action, Manning Publications Co., Stamford, CT, USA, 2011.
- http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor
- http://akka.io/
- http://actor-framework.org/
- http://tutorial.ponylang.org/