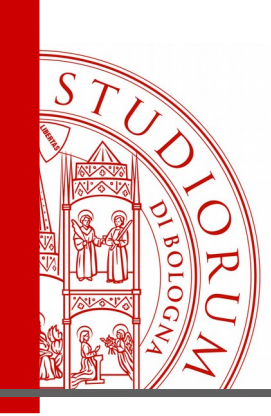


Emerging Programming Paradigms

–

01: Programmazione ad Attori e Erlang

Programmazione ad attori

Programmazione ad Attori

Attore = PID + mailbox + behaviour

- PID = identificatore univoco
- Mailbox = coda per la ricezione di messaggi
- Behaviour mappa messaggi in lista di azioni + nuovo behaviour
- Azione = computazione interna + invio asincrono di messaggi verso un PID + creazione di attori

Programmazione ad Attori

Altro sugli attori

- Un attore corrisponde a un unico thread di computazione
- Esempio di programmazione reattiva/event driven: la computazione avviene solo come risposta a un messaggio
- Gli agenti NON condividono stato, memoria, ...

Programmazione ad Attori

Sistemi di attori

- Un sistema è composto da più attori in esecuzione
- L'esecuzione è indipendente dalla locazione fisica degli attori (i PID sono univoci sull'intera rete) e la topologia è variabile (attori nascono, muoiono, ...)
- Più attori possono trasparentemente essere eseguiti sullo stesso nodo/virtual machine/core/CPU/PC/… continuando a non condividere informazioni, stato, memoria

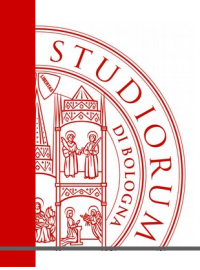
Linguaggi ad Attori

Prima proposta 1973, Hewitt, Bishop, Steiger, “A Universal Modular Actor Formalism for Artificial Intelligence

- Linguaggi accademici: moltissimi
- Linguaggi “mainstream”:
  - Erlang (1987), non influenzato da precedenti linguaggi ad attori
  - Elixir: implementato sulla VM di Erlang (BEAM)
  - Recentemente: numerose librerie per altri linguaggi Java/Scala (Akka), .NET, C, Rust, Haskell, Swift, Groovy, Python, Clojure, F#, C++, Ruby, Lua, Go, ...

Storie di successo

- Akka (Java): Linkedin, Verizon, Intel, Samsung, Unicredit, Twitter, PayPal, Zalando, Verizon, gaming companies, …
- Erlang: Ericsson (AXD301 switch was available 99.999999% of the time!), Amazon, Yahoo!, Facebook, WhatsApp, Motorola, ejabberd, ...



Erlang

Erlang: obiettivi

- Sviluppato presso la Ericsson per implementare software di telecomunicazione (switch telefonici, …)
- Leggere: “A History of Erlang”, Joe Armstrong
- Designed to run concurrent programs that RUN FOREVER
- Concorrenza massiva (>= 100000 attori), algoritmi distribuiti, soft real time, software upgrades frequenti, fault tolerance, very large networks, hardware interaction, on-the-fly code update

Erlang: influenze

Erlang = Concurrent Functional Programming Language

- Linguaggi concorrenti: Ada, Modula, Chill
- Linguaggi funzionali: ML, Miranda
- Sintassi: ispirata da Prolog, usato per prime implementazioni insieme a Smalltalk

Erlang: design decisions

- No memoria condivisa, lock, mutexes,
  - Gestione troppo complessa
  - Non funziona in scenari distribuiti
  - Pessimo per fault tolerance
- Message passing asincrono, ricezione out-of-order
  - L'ordine è casuale in scenario distribuito
  - Se un messaggio viene estratto dalla mailbox al momento errato deve essere processato esplicitamente più tardi

Erlang: design decisions

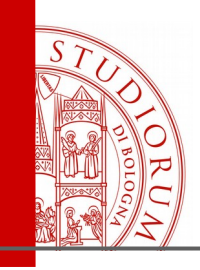
- Fault tolerance via “Let it fail!”
  - Gestione degli errori complessa in scenario distribuito (n processi devono cambiare stato, processi remoti possono essere irraggiungibili, deadlock se messaggi persi, …)
  - Let it fail:
    - in presenza di errori, si uccide un processo e tutti coloro strettamente a lui connessi
    - un supervisore fa ripartire i processi terminati

Erlang: storia @ Ericsson

- 1982-1985:
  - Provati > 20 linguaggi per programmazione di infrastrutture di telecomunicazione
  - Conclusione: very high level PL per mantenere alta la produttività (Lisp, Prolog, …)
- 1985-1986:
  - Esperimenti con Lisp, Prolog, Parlog, …
  - No Lisp: primitive per comunicazione e fault tolerance
  - No Prolog: backtracking è male
  - No Parlog: 1-1 process mapping

Erlang: storia @ Ericsson

- 1987-1992:
  - Sviluppo di Erlang
  - Rapida adozione da parte di altre telcom (e.g. Bellcore)
  - Distributed runtime tardivo
  - Erlang/OTP (set di librerie/framework per lo sviluppo di applicazioni distribuite)
- 1993:
  - Ericson forma divisione per lo sviluppo e la vendita di Erlang, con supporto per hardware eterogeneo



Erlang: storia @ Ericsson

- 1993-...:
  - Ericson decide di non usare più Erlang
  - Erlang sviluppato da altre compagnie
- 1998:
  - Erlang rilasciato come open source
- 2006:
  - Aggiunto supporto a multi-core
  - Nessun cambiamento al linguaggio!
  - Implementazione significativamente cambiata

Erlang: retro-prospettiva

1986: linguaggio logico + concorrenza

1995: linguaggio funzionale + concorrenza

2005: linguaggio ad agenti con agenti scritti in un linguaggio funzionale

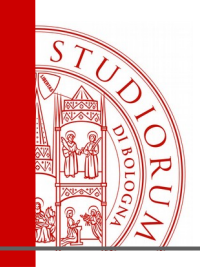
La scelta di implementare gli agenti in un linguaggio funzionale non è obbligata (vedi Akka per Java, ...)

Erlang oggi

- Virtual machine: BEAM
  - BEAM gira come unico kernel process multi-threaded
  - 1 kernel thread per core
  - 1 attore = 1 language thread, schedulati sui kernel thread con automatic load balancing
  - Context switch per i language thread estremamente lightweight
  - Memory footprint dei language thread estremamente lightweight (circa 300 word)
  - Supporta fino a milioni di language thread (e.g. in WhatsApp)
  - Hot code swap: possibilità di caricare a run-time una nuova versione del codice

Erlang oggi

- Libreria standard: OTP
  - OTP = Open Telecom Platform
  - Client e server generici
  - Monitors
  - Finite state machines
  - Distributed database server (Mnesia)
  - Interfacciamento con COM/CORBA
  - ...



Elixir

elixir

- Elixir è un linguaggio ad attori implementato su BEAM
- Sintassi basata su Ruby (Erlang è più simile a Prolog)
- Aggiunge a Erlang
  - metaprogrammazione/macro igieniche (e.g. per implementazione di DSLs)
  - Polimorfismo via protocolli (~ interfacce)
  - ...

Erlang: il linguaggio

Risorse per imparare Erlang

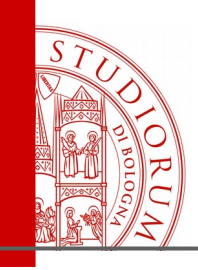
http://erlang.org

In particolare:

- Learn you some Erlang for real good!

Vedi anche:

- http://www.it.uu.se/edu/course/homepage/avfunpro/ht17



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Claudio Sacerdoti Coen

Dipartimento di Informatica: Scienza e Ingegneria (DISI)

claudio.sacerdoticoen@unibo.it

www.unibo.it