ACTORBASE

INGEGNERIA DEL SOFTWARE

Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica, A.A. 2015– 2016

rcardin@math.unipd.it

SOMMARIO

- Go reactive!
- NoSQL database
- Actor model
- Actorbase

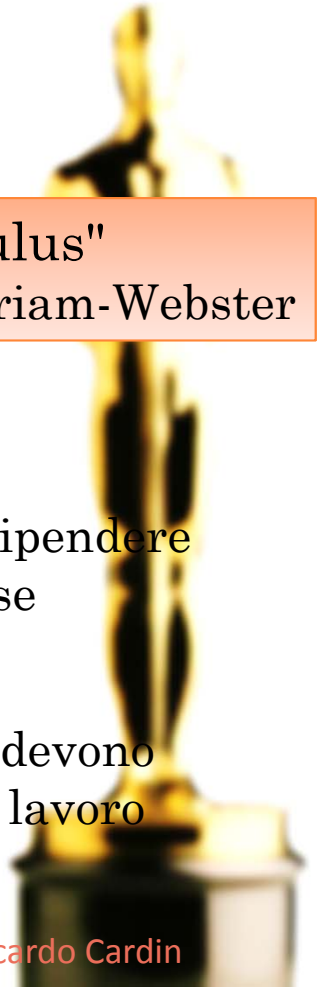
Go reactive!

- Applicazione anni 2000
  - Decine di server
  - Tempi di risposta nell'ordine dei secondi
  - Ore di *downtime* per manutenzione
  - Dati nell'ordine dei Gigabyte
  - Accedute da dispositivi *desktop*
- Applicazione moderne (≥ 2010)
  - Cluster di migliaia di processi *multicore*
  - Tempi di risposta nell'ordine dei millisec
  - 100% *uptime*
  - Dati nell'ordine dei Petabyte
  - Accedute da qualsiasi tipo di dispositivo

GO REACTIVE

- Nuovi requisiti richiedono nuove tecnologie
  - *Reactive Application*
    - Orientate agli eventi
    - Scalabili
    - Resilienti
    - *Responsive*

"Readily responsive to a stimulus"
Merriam-Webster



react to events
La natura *event-driven* abilita alle altre qualità

react to load
La scalabilità non deve dipendere da risorse condivise

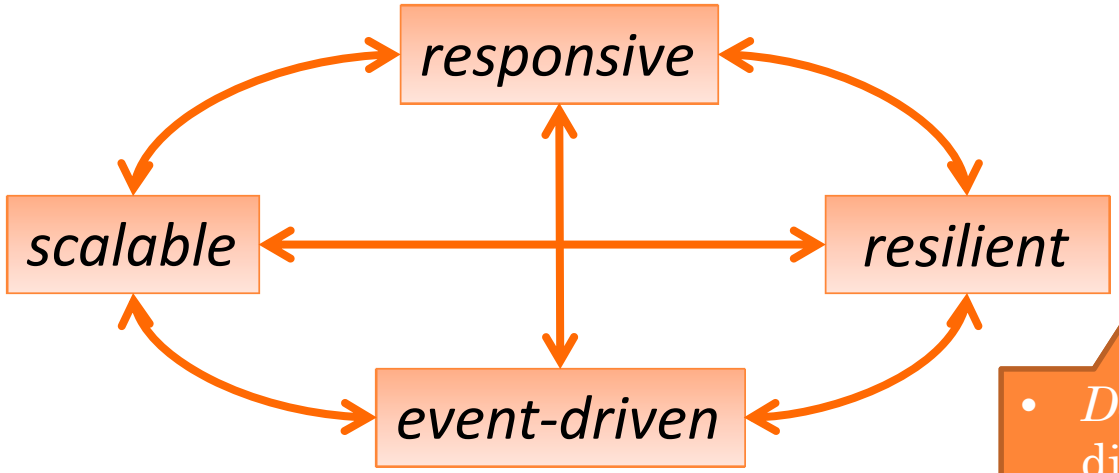
react to failure
Sistemi resilienti permettono di recuperare errori a tutti i livelli

react to users
I tempi di risposta non devono dipendere dal carico di lavoro

REACTIVE MANIFESTO

- Nessun *redesing* per ottenere la scalabilità
- Scalabilità *on-demand*
- *Risk-management*

- *Real-time, engaging, reach, collaborative*
- Nessuna latenza nelle risposte



- *Downtime* è perdita di denaro
- Parte del *design*

- *Loosely coupled design*
- *Communication orientation*
- Uso efficiente delle risorse

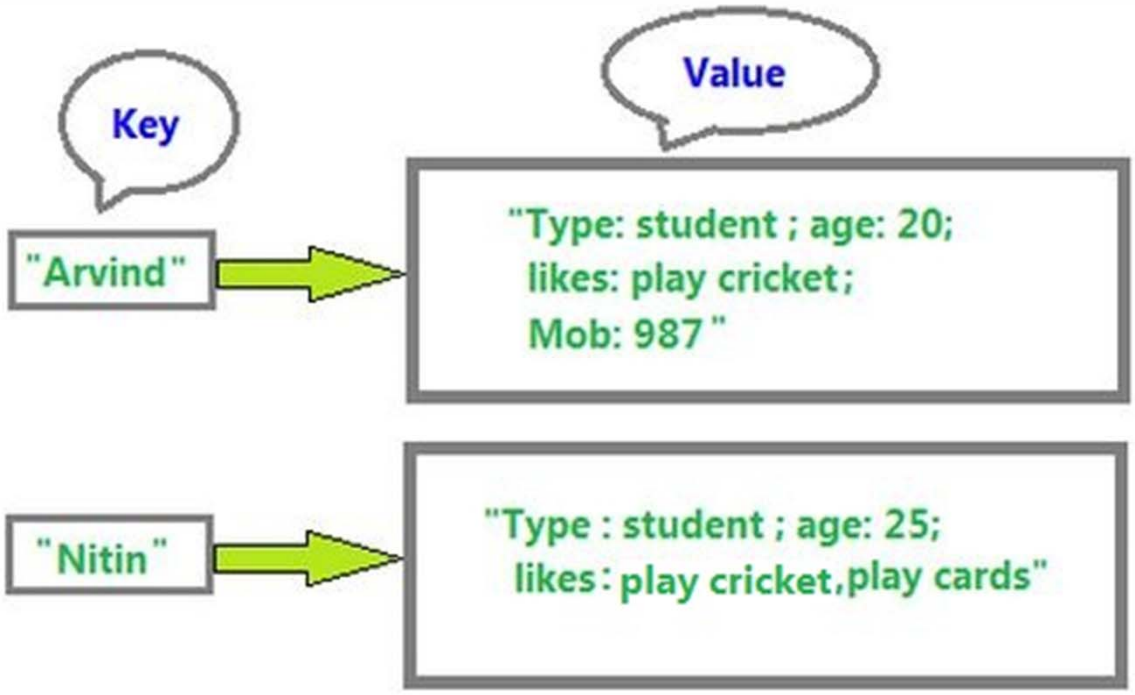
NoSQL Database

- Applicazione *reactive* richiedono un nuovo paradigma di accesso ai dati
  - Non è più possibile garantire le proprietà transazionalità (ACID)
    - Atomicità
    - Coerenza
    - Isolamento
    - Durabilità / persistenza
- NoSQL: «Not only SQL»
  - Permettono la gestione dei casi nei quali il modello relazionale rappresenta una forzatura

NoSQL Database

- Lo schema di persistenza individua il tipo di DB
  - Document oriented: MongoDB
  - Graph database: Neo4j
  - Column oriented: Apache Hbase, Apache Cassandra
  - Key-value map: Amazon Dynamo
- CAP theorem
  - Non è possibile garantire contemporaneamente *coerenza*, *disponibilità* e *partizionamento*
    - Al massimo due proprietà su tre
- Key-value: array associativi (mappe / dizionari)
  - Dati strutturati o meno

KEY-VALUE DATABASE



- *Scaling* orizzontale
  - Aggiunta trasparente di nodi al sistema distribuito
    - L'elaborazione viene ripartita tra i nodi

ACTOR MODEL

> Each actor is a form of reactive object, executing some computation in response to a message and sending out a reply when the computation is done
>
> John C. Mitchell

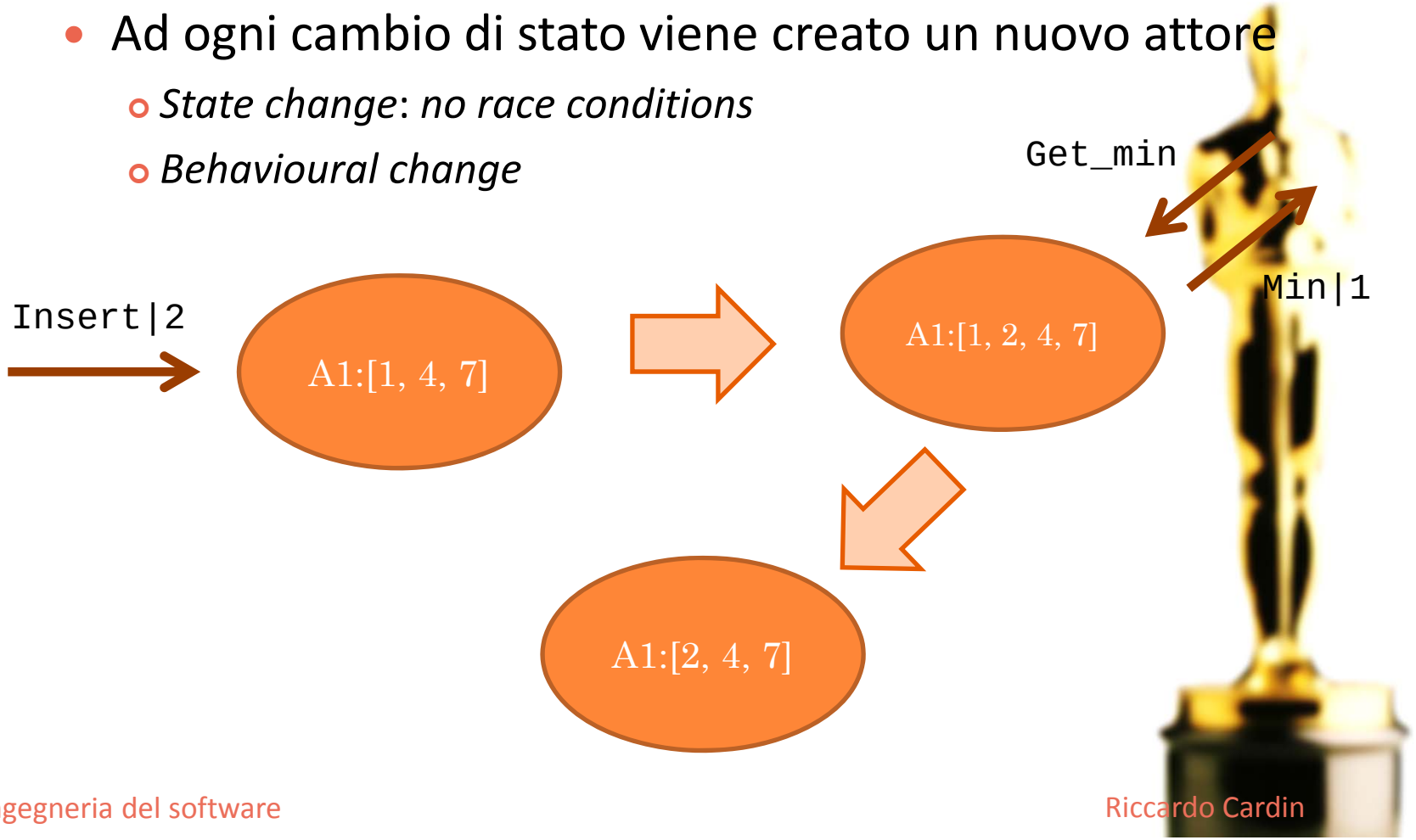
- *Reactive*
  - Elaborazioni solo in risposta a stimoli esterni
    - Dormiente / attivo
    - Non c'è un concetto esplicito di *thread*
- Tre azioni fondamentali
  - Invio di un messaggio async a se stesso o altro attore
  - Creazione nuovi attori
  - Modifica del proprio comportamento
    - Nessuno stato interno (...in teoria...) → Immutabili

ACTOR MODEL

- Messaggi (*task*)
  - Interfaccia di comunicazione dell'attore
    - Variare nel tempo (comportamento)
  - Elaborazione dei messaggi uno per volta
    - Coda di messaggi (*mail box*)
    - Nessuna garanzia sull'ordine di arrivo
  - *Mail system*
    - Ogni attore ha associato un *mail address*
  - Composti di tre parti

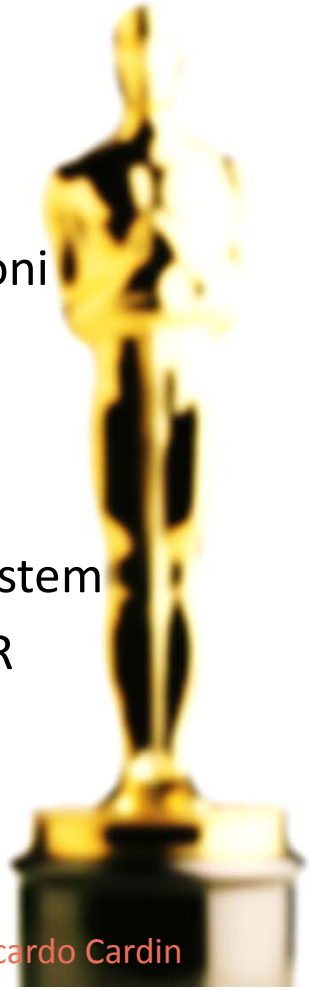
ACTOR MODEL

- Esempio
  - Ad ogni cambio di stato viene creato un nuovo attore
    - *State change*: *no race conditions*
    - *Behavioural change*

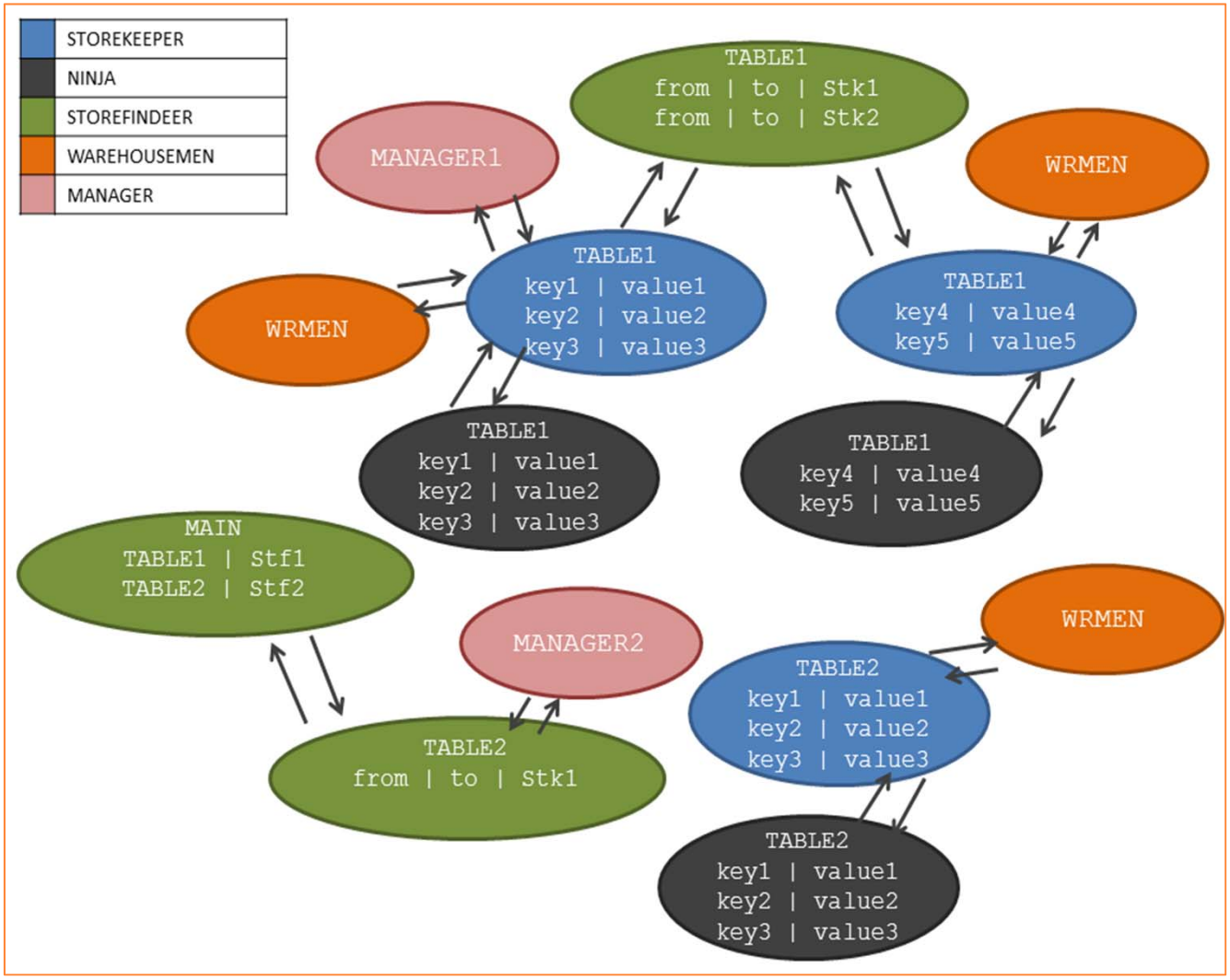


ACTORBASE

- Key-Value database basato sul modello ad attori
  - Ad ogni mappa è associato un nome
  - Le componenti sono singoli attori
    - STOREKEEPER: possiedono le mappe con le informazioni
    - NINJA: *disaster recovery*
    - STOREFINDER: instradano le richieste (indici)
      - MAIN: riceve le richieste dall'esterno
    - WAREHOUSEMEN: persistono le informazioni su filesystem
    - MANAGER: gestiscono le proprietà degli STOREKEEPER



ACTORBASE


STOREKEEPER
NINJA
STOREFINDEER
WAREHOUSEMEN
MANAGER
TABLE1
from | to | Stk1
from | to | Stk2
MANAGER1
WRMEN
TABLE1
key1 | value1
key2 | value2
key3 | value3
WRMEN
TABLE1
key4 | value4
key5 | value5
TABLE1
key1 | value1
key2 | value2
key3 | value3
TABLE1
key4 | value4
key5 | value5
MAIN
TABLE1 | Stf1
TABLE2 | Stf2
MANAGER2
WRMEN
TABLE2
key1 | value1
key2 | value2
key3 | value3
TABLE2
from | to | Stk1
TABLE2
key1 | value1
key2 | value2
key3 | value3

ACTORBASE

- Requisiti obbligatori
  - Implementazione sottoinsieme di attori
    - STOREKEEPER
    - STOREFINDER
    - WAREHOUSEMEN
  - Implementazione operazioni base
    - Inserimento
    - Cancellazione
    - Aggiornamento
  - *Domain Specific Language*
    - Riga di comando
  - Pubblicazione su Github

ACTORBASE

- Requisiti opzionali
  - Implementazione tipi di attori rimanenti
  - Driver Scala / Java (JDBC?)
- Tecnologie
  - Scala (consigliato) / Java
  - Akka: actor model on JVM