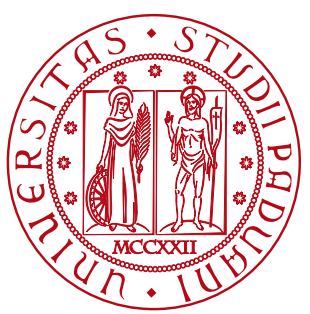
Elixir concurrency model

Flipped classroom
Runtimes for Concurrency and Distribution

Elia Pasquali

06/11/2023



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Definition of scalability

Scalability is the ability of a system to handle a workload that can change during time, managing its resources by applying cost-effective strategies for extending system's capabilities.

A scalable system should:

- guarantee no performance loss when *demand* (users) or *complexity* (nodes) increases.
- when the system is distributed the *state should remain consistent* and identical between nodes at all times.
- be *easy to operate* on for humans at any size of it.

Language characteristics

- Functional approach
- Concurrent and scalable
- Erlang compatible (runs on BEAM, the Erlang VM)

Concurrency feature

- Actor model:
  - *Shared nothing concurrent programming*
  - Message passing
- Lightweight processes
  - Not OS threads
- Fault tolerant:
  - *"Let is crash"* philosophy
- `Agents` and `Tasks` for particular operation

Is it a good model?

- **Performance**: lightweight processes, hundreds of thousands concurrently use all machine resources efficiently (*vertical scaling*)
- **Cost**: *spawning* new processes is an easy and low cost operation
- **Operability**: the system will “regulate” itselft on certain events. This is achieved with fault tolerance and message passing
- **Usability**: since it is based on the Erlang VM, serving an amount of user that could shrink and grow in any moment its not a problem
- **Replica consistency**: messagge passing and the functional approach the shared state is minimized