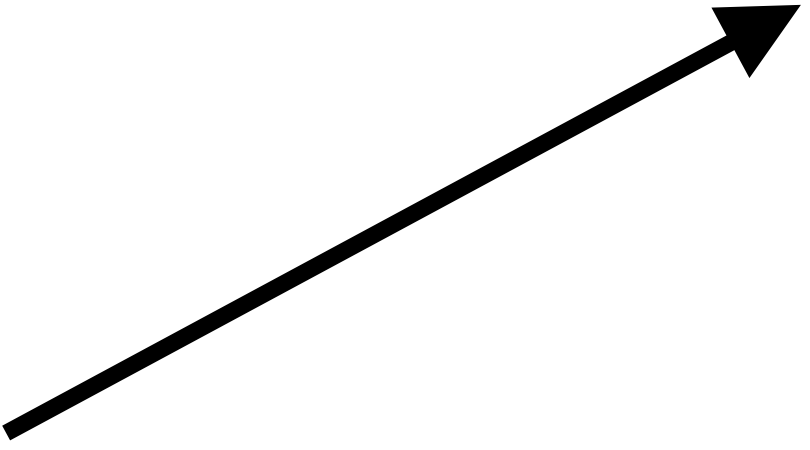
Actor System

Actor = process

- **Can receive messages**
- **Send messages**
- **Spawn new actors**
- **Make decisions about next steps (internal state)**

```haskell
computeParallel :: [Term] -> IO [CTerm]
computeParallel terms = do
  traverse (\term -> actorAsync $ krivineMachine term)
  awaitResults
```



11

- create actor that will do computation
- Send message to that actor, with term
- await results of computations and return them

Evaluation

- **Each case run 10 times and median is taken**
- **PC: 16gb RAM, Intel core I7 - 8 logical cores**

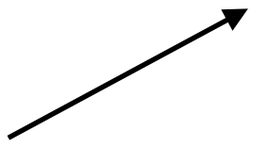
- Raise **n** to the power of **m** and **m** to the power of **n**
  - *((λxλy((a)(x)y)(b)(y)x)λfλz(f)^{n}z)λfλz(f)^{m}z*
    Produces big term. Grows exponentially.
    Could be computed in 2 threads
- Compute something equal in 1..n threads
  - *(λy(..(x)y)..)y)(λxx)^{n}z*
    Big term that produces small result. We compute
    In m threads a lot of identities (Id x = x; id(id(id…))
- Bigger term = more time to serialise/deserialise

Actor System

Actor = process

- **Can receive messages**
- **Send messages**
- **Spawn new actors**
- **Make decisions about next steps (internal state)**

```haskell
computeParallel :: [Term] -> IO [CTerm]
computeParallel terms = do
  traverse (\term -> actorAsync $ krivineMachine term)
  awaitResults
```



- **create actor that will do computation**
- **Send message to that actor, with term**
- **await results of computations and return them**