Multi-core programming

Project Erlang

Twitter

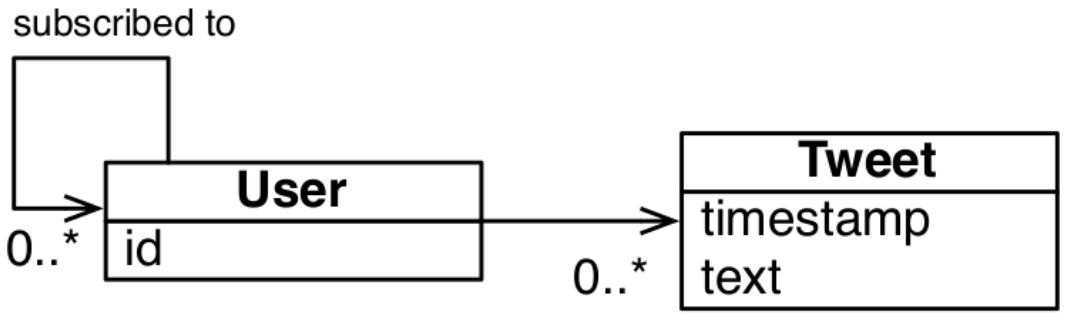
Goals

Scalable Twitter-like service

- Implementation
- Evaluation
- Report

Implementation

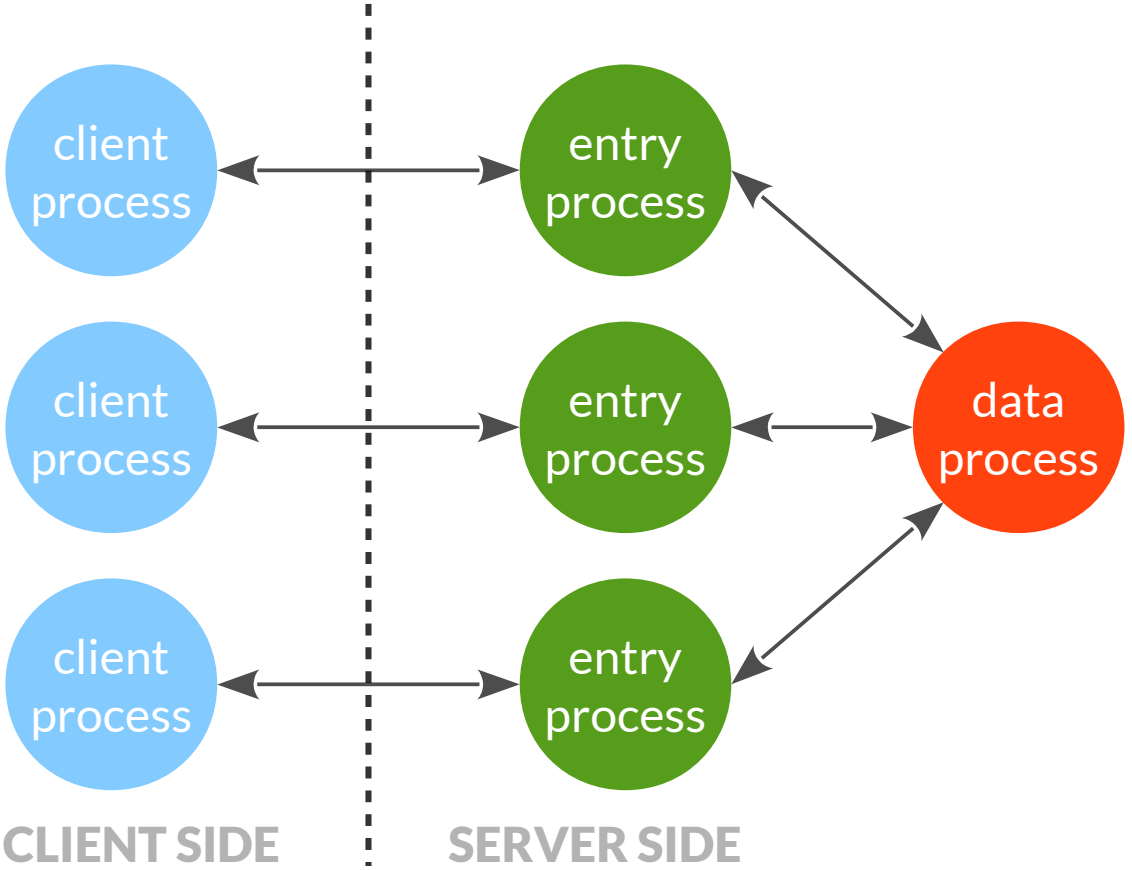
Data model:



Operations:

`tweet`
`get_tweets`
`get_timeline`

Example implementation



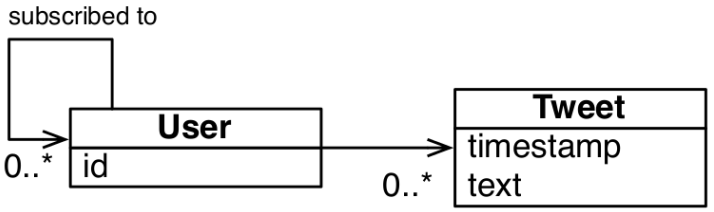
- 1 process / client → don't change, for benchmarking
- 1 entry process = 1 client process } implementation details, free to change
- 1 data process

Scalability over consistency

Allowed to sacrifice consistency for scalability

E.g:

- some requests might return stale data
- keep both subscriptions and subscribers, both lists sometimes inconsistent



In report: describe motivations

In evaluation: measure scalability and (in)consistency

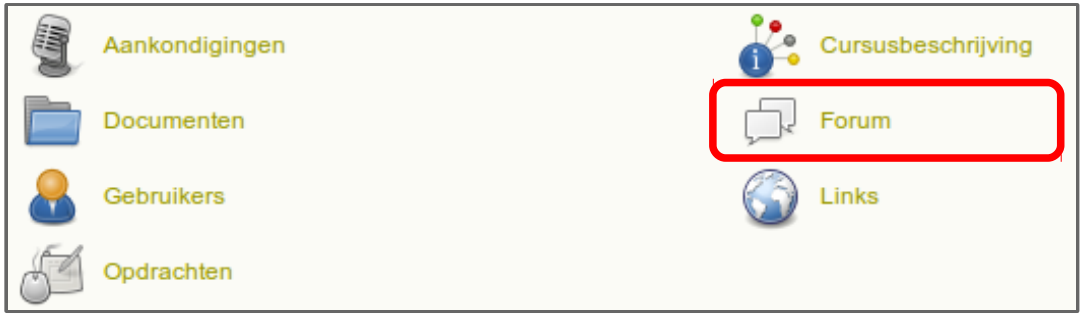
Evaluation

Benchmarks: generate number of client processes, each do a number of operations (`tweet`, `get_tweets`, `get_timeline`)

Latency & throughput of operations, in variety of situations

Best case, worst case, in between

Allowed to share load generation code (only this, nothing else!) → PointCarré forum



Evaluation

Serenity = 64-core server at lab

Time slot of 4 hours (limited!)

Remote log in: demo in lab session

Report may contain experiments on Serenity as well as local (≥ 4 cores)

Report

Table of contents in assignment sheet:

- Implementation
  How did you ensure scalability?
  Where did you sacrifice consistency?
  Include diagrams to show design
- Evaluation
  Describe set-up, metrics
  Include plots of results

Details

Deadline: Sunday 27th of April, 23:59

Submit ZIP with implementation & report on PointCarré

Project defense in June

⅓ of your final grade

Assignment sheet on PointCarré