Load balancing of optimization algorithms

Author: Lukáš Forst
Supervisor: Ing. Ondřej Vaněk Ph.D.

Motivation

- large-scale optimization problems require a significant amount of computational resources
- multiple optimization algorithms should be able to share the same computational resources
- hardware cost reduction

Thesis goals

- formalize load balancing problem of the optimization algorithms
- propose a load balancing algorithm
- design and implement the load balancer

State-of-the-art solutions

- mainly generic
- static/dynamic load balancing
- focused on load balancing of short-running tasks
  - handling HTTP request

⚠ no specific domain knowledge

Formalization

Problem formalization

- formalized as an integer linear programming problem in section 3.1
- specified input/output variables
- resources assignment defined as binary assignment ${}^{r}x_t^j = \{0,1\}$
- various constraints
  - execution cost maximum $\forall t,j:\ P^j \leq C_t^j \implies \sum_{t+1}^{\infty} \sum_{r \in R} {}^{r}x_t^j = 0$
  - execution time maximum $\forall t,j:\ D^j \leq t \implies \sum_{t+1}^{\infty} \sum_{r \in R} {}^{r}x_t^j = 0$
  - resources reallocation $\sum_{r \in R} {}^{r}x_t^j = 1 \implies \forall r \in R:\ {}^{r}x_t^j - {}^{r}x_{t+1}^j \geq 0$
- optimization criteria $\max crit_t = \alpha \sum_{j \in J} S_t^j - (1-\alpha) \sum_{j \in J} C_t^j \qquad 0 \leq \alpha \leq 1$

Proposed load balancing algorithm

Input: Q - queue with jobs to schedule

Q: jobs queue;

J: set of jobs;

P: predictions;

E: execution plan;

$J \leftarrow Q.\text{poll}()$;

$J' \leftarrow J.\text{filter}(job \rightarrow job.D > job.T \wedge job.P > job.C)$;

$J'' \leftarrow \text{convert}(J')$;

$P \leftarrow \text{predict}(J'')$;

$E \leftarrow \text{schedule}(J'', P)$;

$E' \leftarrow \text{convert}(E)$;

Result: Q is empty

Output: E' - execution plan

Algorithm 1: Load balancing algorithm

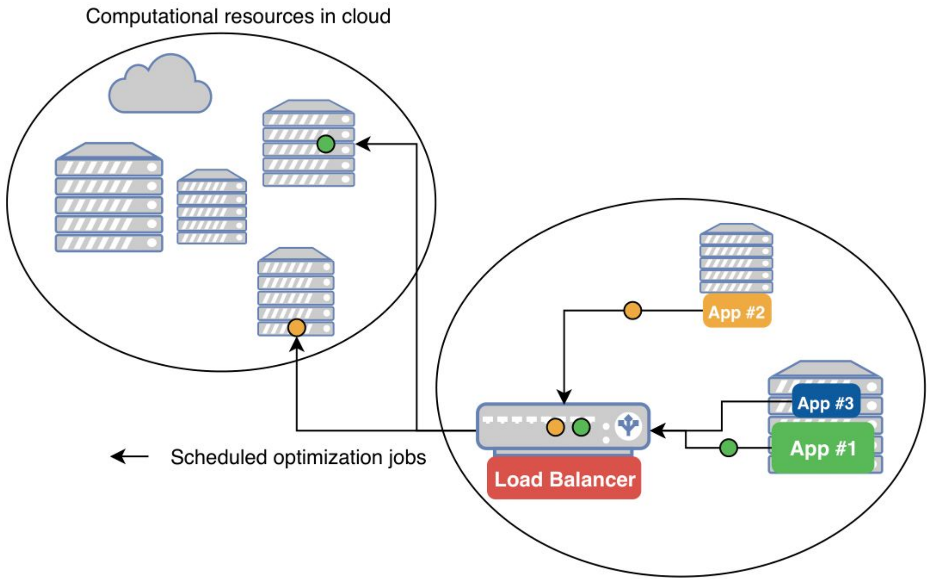
Optimization algorithm

Load balancing decisions

- heuristic algorithm
- OptaPlanner

Algorithm values prediction

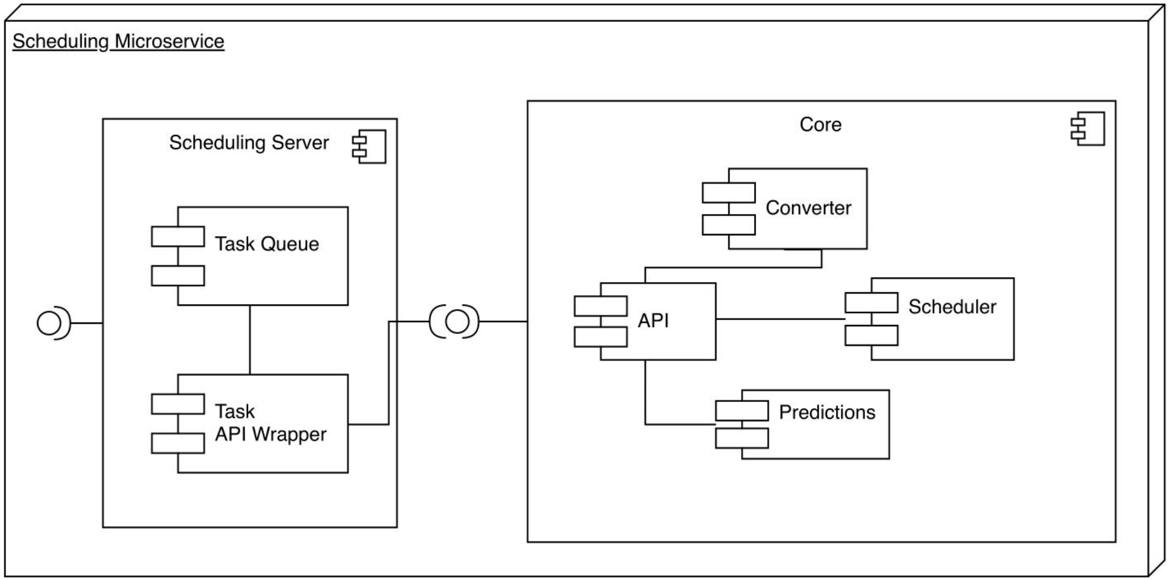
- ${}^{r}\Delta_t^j = {}^{r}|v_t^j - v_{t-1}^j| \cdot {}^{r}x_t^j$
- hyperbola time-series fitting
- Levenberg–Marquardt algorithm



Architecture

Application architecture

- microservice design
- separation of the scheduling core
- core
  - custom data representation
  - optimization engine
- server
  - uses core API
  - exposes scheduling API



Implementation

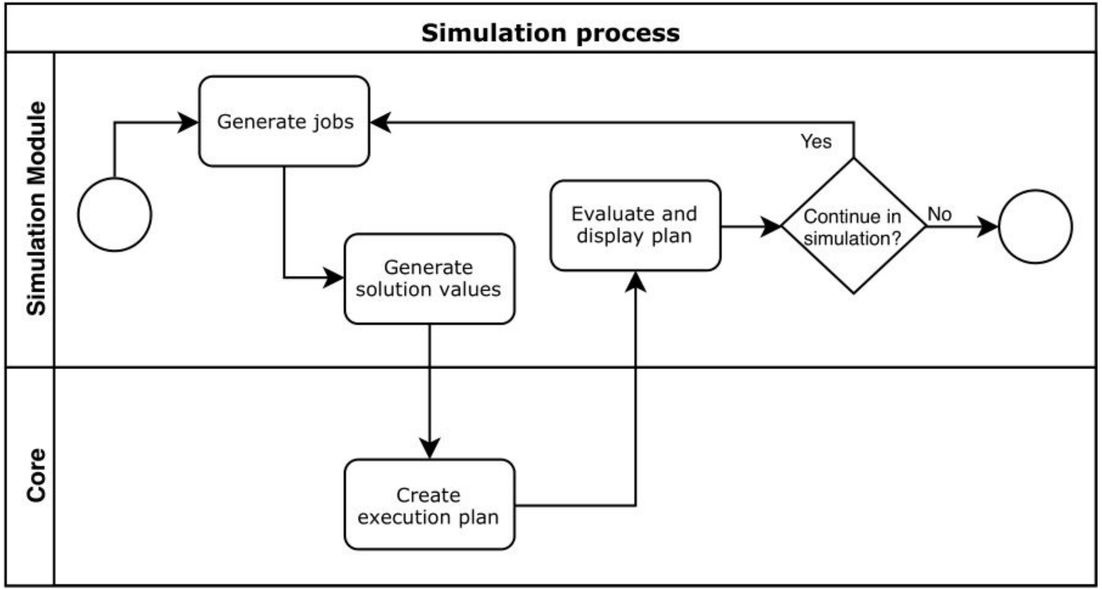
- Java Virtual Machine
- Kotlin
- Docker
- Docker Compose
- Ktor

Routes Discovery Library

Simulations

- implemented in separate module

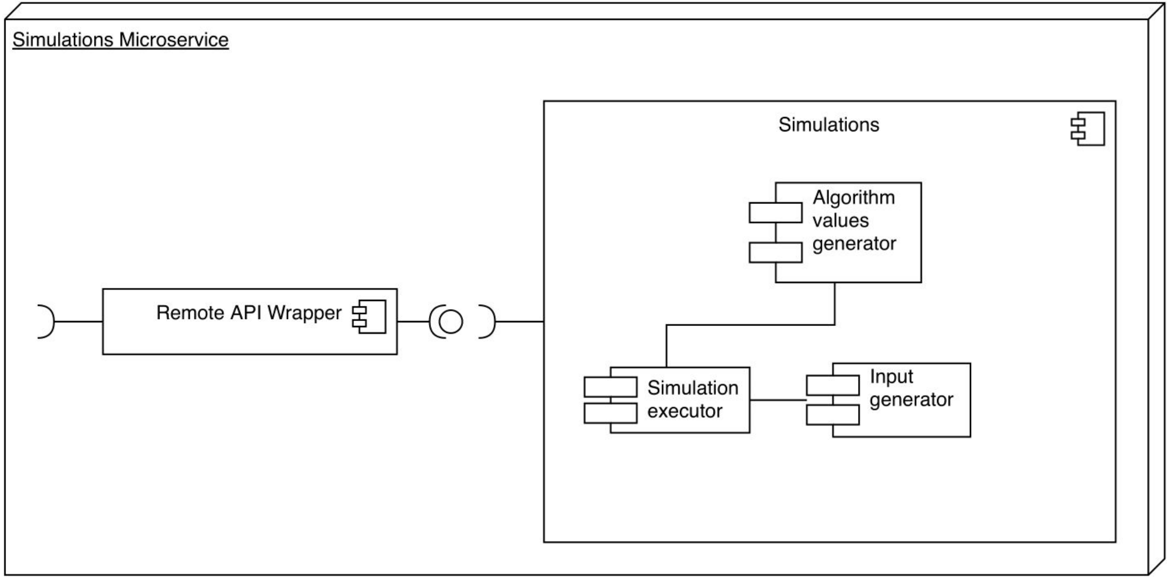
- simulation control
- input generators
- local/remote simulations
- random parameters generation
- objective function values created by instances of heuristics algorithms



Simulations

- simulation microservice
- simulates real-world usage
- uses the same simulation core

- executed inside Docker Compose



Conclusion

Future work

Infrastructure

- execution module development
- optimize resource usage
- fine-tuning of the optimization engine

Routes discovery library

- refactoring
- usage of the generic dependency injection framework
- publish under an open-source license

Conclusion

- traditional load balancing strategies are not optimal
- domain specific load balancing of optimization algorithms
- problem formalization
- design and implementation of the load balancer
- simulations and evaluation of the load balancer

Thank you for your attention!

Author: Lukáš Forst
Supervisor: Ing. Ondřej Vaněk Ph.D.

Jaký je výpočetní overhead load-balancing platformy samotné, na kolik zatěžuje rozvrhovací algoritmus platformu samotnou?

- jedná se o další optimalizační úlohu navíc
  - tedy zatěžuje minimálně jeden zdroj dle definice
  - zatížení pouze ve chvíli, kdy algoritmus plánuje
- algoritmus samotný je paralelizovatelný
  - počet vláken, které může algoritmus použít je vstupem

V jaké škále a od jaké velikosti rozvrhovaných jobů dává platforma smysl?

- klasické techniky rozvrhování zátěže selhávají
- není možné uspokojit všechny minimální nároky algoritmů najednou
  - v jednoduchých případech možné řešit prioritní frontou

$$\exists t : \sum_J j_t^{min} > |R_t| \text{ kde } |R_t| \geq 5$$

Objective function over iterations