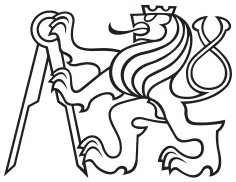

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Unassisted project report

Lukáš Forst

Supervisor: Ondřej Vaněk, Ph.D.
January 2019

Contents

Chapter 1

Introduction

Optimization algorithms and solutions build on them are widely used in current manufacturing industry to reduce production costs. With more and more production automatization, optimization algorithms can manage and schedule whole factories with maximum available efficiency.
Complexity of optimization problems could be huge and therefore performance requirements are sometimes not easily satisfiable. Using one powerful instance of optimization algorithm in cloud seems like a solution for problems with smaller complexity, but what if we have multiple huge problems where each is performance demanding? Of course, we can create multiple instances, but that would be expensive and not well manageable and scalable since adding another instance requires some time. Another disadvantage of this approach is the fact, that optimization algorithm is not running 100% of time and thus resources allocated by this algorithm are unused while other algorithm instances could be potentially overwhelmed. Also paying for unused hardware is wasting money and optimization algorithms are supposed to save money.
Now imagine having two completely different problems that each requires its own application which visualises data and optimization algorithm to compute some kind of plan, this algorithm can be generic enough to operate on both domains with same code base, but it requires a lot of performance resources. If we use monolithic architecture of both applications, we would have same code in two applications, but what is even worse, we would need two powerful machines to run our applications. As previously mentioned, these two machines would not be using their power whole time and would be mainly idle.
What if one application runs only few minutes a day, but needs that power to complete tasks in time? A lot of resources would be wasted if it has its own server, but using not powerful server would lead to increasing duration of ongoing tasks which is something we do not want.
Solution might be **load balancer** specifically developed for optimization algorithms.

1.1 Problem definition

1.1.1 What I want

1.1.2 Remote scheduler server requirements

1.2 Motivation to solve it

Chapter 2

Technical Background

2.1 Algorithms

This work does not contain any own algorithm implementation for scheduling problems, instead I would like to use pre-prepared and already implemented solver. First we must specify which kind of approach we would like to choose. We have many options, how to represent and then solve scheduling problem such as

- Linear programming
  - more specifically mixed integer programming
- Constraint programming
  - more specifically heuristics algorithms

2.1.1 Linear Programming

Some general information about linear programming goes here

Advantages of linear programming approach

Disadvantages of linear programming approach

2.1.2 Heuristic algorithms

General information about heuristic

Advantages of heuristic algorithms

The main advantage of heuristic algorithms is that they offer a quick solution for problem they are solving.

Disadvantages of heuristic algorithms

The main downside of HA is the fact, that they can't guarantee that found solution is the optimal one.

2.1.3 Existing solutions

After considering all advantages and disadvantages of previously mentioned approaches, I decided to use *heuristic algorithm* in my implementation of RMS. Since it is not main goal of this work to implement such a algorithm I will choose some existing implementation of some heuristic algorithm which is general enough to be used in my paper.
I would like to present two selected existing implementations

- OptaPlanner
- TASP

OptaPlanner

OptaPlanner is generic heuristics based constraint solver.

TASP

Task and Asset Scheduling Platform is a lightweight framework developed by Blindspot Solutions designed to solve a large variety of optimization and scheduling problems from the area of logistics, workforce management, manufacturing, planning and others. It contains a modular, efficient planning engine utilizing latest optimization algorithms. TASP is delivered as a software library to be used through its API in applications which require powerful scheduling capabilities.

2.1.4 Selected algorithm

I decided to use TASP in for my paper, because ...

2.2 Load Balancing

There will be some info about how should server balance itself.

- prioritisation - mainly done by priority queues
- handover
- instance sizing
- algorithms - following are methods used in network balancing -> probably can't be used because we need to manage scheduling which is heavy on computer resources like CPU/RAM/IO
  - The Least Connection Method
  - The Round Robin Method
  - The Least Response Time Method

Chapter 3

Related Technologies

In this section I would like to mention related technologies that could be potentially used while implementing described RMS.

Bibliography