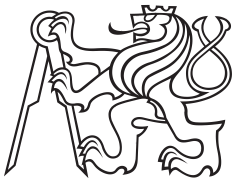

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Unassisted project report

Lukáš Forst

Supervisor: Ing. Ondřej Vaněk, Ph.D.
January 2019

Contents

Chapter 1

Introduction

1.1 Problem definition

1.1.1 What I want

- I have generic scheduling algorithm which does not entirely need to know the domain it is operating with
  - It has its own data representation
- I have many different domains where each requires its own visualisation application (let's call it *client* application)
- I want to have one instance of scheduling algorithm which can perform scheduling for each client application
  - This instance must be easily scalable and generic such as adding new problem domain is not *"a big deal"*
- I want to change as little as possible when adding a new domain (and thus new client application)
- I want this algorithm to run on dedicated server

1.1.2 Remote scheduler server requirements

- Client application does not contain any scheduling algorithm
  - Client contains converter (which is used to translate data from client's database model into remote scheduler data model) and system which ensures connection with remote scheduling server
- When client wants to create new plan, it just transforms database data into scheduler data model and sends them to **RMS**[1]
- RMS is modified as little as possible while adding new client application (and thus new problem domains)

[1] **R**emote **S**cheduler **S**erver

  - Core scheduling algorithm **must not** be changed
  - No code can be changed when adding new domain
    - Only new domain-specific behaviour can be added such as heuristics, plan evaluation etc.
    - It must be ensured that these domain specific parts of RMS will be used only while scheduling correct domain

- RMS ensures that clients scheduling data/algorithms can't interact with each other
  - Each scheduling must be enclosed and exclusively accessible only for client that started it
  - Domain specific code can't be used in different domain

- RMS is able to balance its load
  - It is possible to outsource SJ[2] to another instance of RMS
  - SJ have priority and assigned resources, when priority is set to low, resources can be reassigned to SJ with higher priority

- RMS supports different scheduling configuration
  - SJ can have different requirements on RAM/CPU/IO
  - SJ can have different complexity and thus different scheduling phases have different performance requirements
    - ie. inserting assignments to final plan, removing assignments from plan, plan evaluation etc.

1.2 Motivation to solve it

- I don't want to include same and unnecessary code into all client applications
  - I don't need whole scheduling algorithm in client's code when data conversion layer would be enough
  - [**MAIN**] I don't want to deploy whole application when scheduling algorithm needs to be adjusted

- I would like to completely separate scheduling core and user service layer
  - Client and RMS communicate on predefined API
  - Client can be implemented in different environment (ie. programming language, runtime environment etc.) than RMS
    - ie. RMS in JVM and client in .NET

[2] SJ - Scheduling Job

        - This is one of the main advantage of RMS, because sometimes, client application must be build on specified technology, if we have one implementation of scheduling algorithm in different technology, than is specified, we would need to reimplement it

- Scheduling is resource-intensive task which requires powerful hardware
    - This powerful hardware could be fully utilized only when scheduling is running
    - It's not always possible to have such a powerful server for each client application to solve complex and large-scale optimization problems
    - If I use powerful machine, majority of power is not used for most of the time, because scheduling algorithm is not running
        - This power can be potentially used elsewhere - for example in applications that, for some reason, can't have powerful server
        - Even when having powerful machine, scheduling task is resource-intensive and therefore server response time could be higher than usual, this usually leads to lowered user experience
    - Different phases of scheduling have different resource requirements
    - If I move scheduling algorithm out of user-serving application, I don't need such powerful hardware for each application
- I want one central place, where all applications perform scheduling
    - I would have control over total resources allocation for scheduling
    - I could assign priorities to SJ
- I want to be able to change resources allocation during scheduling
    - Resource allocation according to scheduling phase

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