Contents

Acronyms

AI Artificial Intelligence. 4–6, 9

KPI Key Performance Indicator. 7

ML Machine Learning. 5, 6, 9

RL Reinforcement Learning. 4–7, 9, 10

Chapter 1

Introduction

The cloud is everywhere. All companies in the tech big 5, i.e. Facebook, Apple, Amazon, Netflix and Google, use the cloud extensively and offer many cloud services to their users. Movies are not bought anymore, but streamed. New MacBooks do not come with much storage, instead buyers get some storage in the Apple cloud. But how are the movies and pictures stored in the cloud? What even is the cloud? In this thesis it is shown that the cloud is a network of many computers. These computers need to be managed, which is done by schedulers. Efficient schedulers is very useful, because this means that using cloud services becomes more efficient. In this thesis we look at how methods for improving this efficiency can be selected.

In this chapter the many aspects of resource schedulers in cloud environments are discussed. Firstly, in Section 1.1 cloud environments are discussed. Secondly some theoretical background about the problem of job scheduling and its NP-hardness is provided in Section 2.2. Lastly, in the Section 1.2 the scope of the presented thesis is defined, the current situation is evaluated, the gap is identified and a research question is stated.

1.1 The Cloud

The cloud is a popular term used for (a network of) data centers containing many computers that provide resources, for example storage and computing power. These computers communicate via standard network protocols locally. Data centers thus contain many computers that provide computing resources. The four major advantages of using data centers for large computing tasks are the following: 1) Distributed. Data centers are the ideal environment for running distributed software, because the nature of data centers is distributed computing; 2) Robustness. When one computer breaks, the task can be sent to another computer and

the broken computer can be replaced while the application is still running; 3) Configurable. The computer can be selected based on the resource-need of the task. Providing many different configurations can make sure that tasks fully utilize all the resources the computer provides; 4) Scalability. Data centers are modular and thus easier to scale. New computers can be added and connected to the network and directly be used.

An additional benefit of the cloud is how companies pay for the resources they need. This way, companies never pay for idle resources and can easily scale up to use more resources when needed. By building and sharing huge data centers, companies can effectively achieve the economies of scale principle for needed resources. This reduces the cost of the resources, but also the maintenance costs. The reason cost is reduced because resources are bought in bulk and maintenance costs are reduced because there are less maintainers needed.

1.2 Current Situation and Improvements

In this section current traditional (non-AI) schedulers are reviewed and improvements using Artificial Intelligence (AI) techniques like Reinforcement Learning (RL) are evaluated. The problem with current AI improvements of schedulers are identified and the aim of this resource is stated. The scope of this thesis is narrowed down to reducing retraining time of RL based schedulers in cloud environments. The choices made in defining the terms and scopes of this thesis are listed in the following paragraph.

1.2.1 Traditional Schedulers

Current cloud resource managers are developed specifically for the system it manages, based on simple heuristics and fine tuned by trial and error. Creating the resource managers is a hard and tedious task. A common aspect of the resource managers based on simple heuristics is the straightforwardness. Current cloud schedulers are developed for ease of understanding. The schedulers generalize, i.e. they perform the same job regardless of whether the workload is heavy or light (Mao, Schwarzkopf, Venkatakrishnan, Meng, & Alizadeh, 2019). Three classic (non-AI) algorithms are explained below to provide an idea of the simplicity of the non-AI schedulers.

1. First in, first out (FIFO): This algorithm treats the awaiting jobs like a queue and lets later jobs wait until earlier jobs finish and resources are available for the next job.

2. Shortest Job First (SJF): This algorithm sorts awaiting jobs based on increasing order of completion time.

3. Tetris: ?

   hier tetris

The above explained algorithms are highly intuitive resource managers and not fine tuned for different workloads. Due to the lack in flexibility of these non-AI algorithms , there are also situations in which they will perform much worse than other algorithms will do. For FIFO this leads to multiple jobs with a long duration time, blocking all other jobs till finish. The disadvantage of SJF is that it can cause starvation, meaning that short jobs are constantly added and will never be executed.

Currently, schedulers are not capable of handling differences in workload and have other shortcomings. There is a need for schedulers that are capable of handling increasingly complex large-scale systems, although the systems are currently already too complex for humans to fully understand and schedule. Due to the shortcomings of current non-AI implementations, previous research is done on implementing these resource managers using AI technology. Recent research, e.g. Mao et al. (2019); Mao, Alizadeh, Menache, and Kandula (2016); Zhang et al. (2020), has shown that using deep reinforcement learning for resource management improves average job completion time by at least 21% (Mao et al., 2019).

1.2.2 The Problem

In the previous section the problem with non-AI schedulers is identified. Thereafter, improvements using RL are shown, but these RL improvements also have their shortcomings. One of these shortcomings is flexibility. A known problem of Machine Learning (ML) algorithms is overfitting on the training environment and not being able to work with environmental changes. This is also the case with RL based schedulers. Schedulers can show undefined behavior when resources are added or removed. Thus, when a change in resources is done, the RL based schedulers need to be retrained to work in the new environment. Retraining is a common problem with ML algorithms. Retraining takes time, in which a sub-optimally working scheduler is still scheduling. Retraining also costs computing resources itself. This costs money and contributes to global warming. Carbon emission of large ML models is a real issue which is currently researched on, e.g. Patterson et al. (2021). Finding ways to reduce retraining time of RL models in cloud environments is important, because many schedulers can benefit from using these techniques. This makes cloud environments more efficient which is better

for the company in terms of costs, efficiency and maintenance and better for the environment. It also contributes to more robust RL schedulers. If RL schedulers are more robust, more cloud companies will switch to using RL based schedulers in stead of the current non-AI schedulers.

The goal of this research is to integrate robustness into current state-of-the-art ML based resource managers. This leads to the following research question: How to select a method for reducing retraining time of reinforcement learning based resource schedulers in cloud environments? This will be answered by the following sub-questions:

1. How can we effectively assess and compare RL based methods in job scheduling?
2. What are the indicators to assess robustness of RL based schedulers?
3. What are the state-of-the-art approaches for reducing retraining time?

state the research aims and/or research objectives

The goal of this research is to provide a method for selecting methods that reduce retraining time. By reducing retraining time the models become more flexible and

TODO: State the hypotheses

Firstly the state-of-the-art RL schedulers and state-of-the-art methods for reducing retraining time for RL algorithms are reviewed and explained. In the following chapter, the methodology of selecting methods is described. Thereafter the results of this research are shown. Lastly, a conclusion of the results is formed and discussed.

Chapter 2

Related Work

In this chapter related work is discussed. In the first section the scope of the thesis is defined and the methodology of finding related work is described. Thereafter related work on the topics of job scheduling and its NP-hardness, RL based schedulers, assessing robustness of RL methods and approaches for reducing retraining time is discussed. Finally, the gap of in the related work is shown.

2.1 Scope and Methodology

Comparing different methods and models is the main task in this thesis. To compare methods of models, a justified way of assessing methods and models is required. It is also important that the comparison is done on the indicators on which the performance depends. These are called the Key Performance Indicators (KPIs) . The KPIs of RL based schedulers are established from literature. Comparing the robustness of a model is also necessary for this research.

unfinished

2.2 Job Scheduling

Imagine cloud systems as described in Section 1.1. These systems have many tasks, or as in this thesis called jobs, that need to be executed for it to function. Jobs are scheduled by a job scheduler or simply called the scheduler. These jobs vary a lot in duration time and resource-need. To illustrate, if the cloud environment provides a back-end for a website, a job could be one of the following: Serving the right HTML page to a visiting user, compressing user uploaded images, spam filtering, detecting anomalies in user logins or many other types of jobs that need to

be executed to provide a fully functioning website. The schedulers distributes jobs over many resources, provided by a cloud environment. Optimally distributing the jobs is important for cloud environments, because efficiently using resources means there are less resources needed and jobs will complete in a shorter time. Thus, well working schedulers are important for cloud environments, because efficiently using resources saves money and is easier to maintain.

What is efficient and how can schedulers be efficient? There are many metrics on which optimization can be done to make a scheduler more efficient. For example minimizing job slowdown time, minimizing average completion time, maximizing throughput (jobs per time unit) or minimizing total completion time (the makespan). Schedulers are created to be optimized on one or more of these aspects, varying the importance. Unfortunately, the optimization of schedulers is complex. It is a well-known problem in computer science, because of its NP-hardness. In the next paragraph is explained what it means to be a NP-hard problem and the reason that job scheduling is NP-hard.

2.2.1 NP-Hard

NP-hardness is a term used in the P versus NP problem. This problem is one of the seven Millennium Prize Problems (Cook, 2006) and still unsolved. The P versus NP problem is about computational complexity, a way of categorizing problems based on 'how hard' they are. Computational problems have two complexity aspects: the complexity to solve the problem and the complexity to verify if the solution to the problem is correct. The question is if the solution to a given problem can be verified quickly (in polynomial time), is there an algorithm that can find the solution quickly? If this is true, P equals NP. Many computer scientists believe this is not the case for all problems (Rosenberger, 2012). Believed is that there are problems that can be verified quickly but not solved quickly, which are called NP-hard problems. A well-known NP-hard problem is the Sudoku puzzle, especially larger ones (Yato, 2003).

Job scheduling is a generalized version of the traveling salesperson problem. The problem is as follows: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" (Flood, 1956). In this problem, the cities are the resources and the salesman is the job.

Job scheduling is NP-hard because it can be derived from the Graph Coloring Problem, as done in Karp (1972). The graph coloring problem is a NP-hard problem itself.

2.3 Reinforcement Learning

Most ML based schedulers are RL based, and therefore this thesis focuses on reducing retraining of RL models. The reason most ML based schedulers are RL based is because of how RL works and differs from the other ML paradigms. RL and its differences with the other paradigms is very well explained in 'the most popular artificial intelligence textbook in the world'[1], Russell and Norvig (2010). Their explanation of RL is summarized in the following paragraph.

RL is one of the basic paradigms in ML , along with supervised learning and unsupervised learning. RL is different from the other two paradigms. Supervised learning and unsupervised learning have a thing in common: the need for data. Supervised learning needs annotated data, various inputs and desired outputs are given and the algorithm learns a function to get as close to the wanted outputs as possible given the inputs. With unsupervised learning a model is forced to build an internal representation of the world by mimicking the data. The reason RL is different is that it does not depend on data, but rather learns from a feedback loop of rewards or reinforcements. It typically consists of one or more agents, an environment, a set of actions and a set of states. This agent performs actions in the environment to reach states. An agent receives a reward for taking actions. In many complex environments RL is the only feasible way to train models because there might be little data available or the environment is too complex to model (Russell & Norvig, 2010). For the reason that RL has no need for input data but solely needs an environment, actions and a reward function it is widely used in research about using AI in resource managers. The popularity of RL and the nature of how RL works is the reason this research is narrowed down to reducing retraining of RL schedulers.

2.4 Evaluation of Reinforcement Learning Models

Currently there is not yet a standard evaluation method for RL models. Due to the absence of a standard evaluation model, reproducibility often becomes more difficult. Work is done in defining standard evaluation methods, but these methods are not yet the industry standard. Three metrics were found. The first evaluation is regret. Regret can only be used as a metric when an agent with optimal policy can be defined. The difference in actions taken is the regret per action. This is defined as the reward for the action of the optimal agent minus the reward for the taken action. Knowing the optimal policy is not always possible. The second evaluation metric is not used literature but rather in practice. Some non-scientific metrics

[1] According to this blog, but also shown on the homepage of the book (http://aima.cs.berkeley.edu/).

that were used to evaluate RL models are popular, well-defined environments like the openAI gym (Brockman et al., 2016). A comparison of RL models is done on their leaderboard[2]. The used metric is the number of episodes it took to solve the problem. Lastly, proposed methods in literature are for example a framework for evaluating RL in Khetarpal, Ahmed, Cianflone, Islam, and Pineau (2018) and a novel evaluation method in Jordan, Chandak, Cohen, Zhang, and Thomas (2020).

2.5 Assessing Robustness of Reinforcement Learning

It is important to use an accurate metric when determining the robustness of RL based schedulers. Numerous methods for adding in robustness are proposed, for example adding "disturbance" to the training environment (Morimoto & Doya, 2005).

NOTE: What are the indicators to assess robustness of RL based schedulers?

NOTE: What are the state-of-the-art approaches for reducing retraining time?

2.6 Approaches to Reduce Retraining Time

2.6.1 Meta Reinforcement Learning

2.6.2 Transfer Learning

2.6.3 Deep Q-Learning

2.7 Reducing Retraining of Schedulers

Gap:

In Chapter 1 multiple RL based schedulers were discussed. These schedulers lack robustness: flexible environments are not handled well. Methods for improving robustness of RL models exist, but are not yet integrated in RL based schedulers. Integrating these methods would improve robustness and reduce retraining time.

[2] github.com/openai/gym/wiki/Leaderboard

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym.* Retrieved 2021-05-06, from https://arxiv.org/abs/1606.01540

Cook, S. (2006). P versus NP Problem. In J. Carlson, A. Jaffe, & A. Wiles (Eds.), *The millennium prize problems* (pp. 87–106). Providence, RI: American Mathematical Society.

Flood, M. M. (1956). The traveling-salesman problem. *Operations research*, *4*(1), 61–75. Retrieved from https://www.jstor.org/stable/pdf/167517.pdf

Jordan, S., Chandak, Y., Cohen, D., Zhang, M., & Thomas, P. (2020, July). Evaluating the performance of reinforcement learning algorithms. In *International conference on machine learning* (Vol. 119, pp. 4962–4973).

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer. doi: 10.1007/978-1-4684-2001-2_9

Khetarpal, K., Ahmed, Z., Cianflone, A., Islam, R., & Pineau, J. (2018). Re-evaluate: Reproducibility in evaluating reinforcement learning algorithms. In *International conference on machine learning.*

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016, November). Resource Management with Deep Reinforcement Learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (pp. 50–56). Atlanta, GA: ACM. doi: 10.1145/3005745.3005750

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019, August). Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication* (pp. 270–288). Beijing, China: ACM. doi: 10.1145/3341302.3342080

Morimoto, J., & Doya, K. (2005, 02). Robust Reinforcement Learning. *Neural Computation*, *17*(2), 335-359. Retrieved from https://doi.org/10.1162/0899766053011528 doi: 10.1162/0899766053011528

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., ... Dean, J. (2021). Carbon emissions and large neural network training. *arXiv:2104.10350 [cs]*. Retrieved 2021-05-04, from https://arxiv.org/

abs/2104.10350

Rosenberger, J. (2012, May). P vs. NP poll results. *Communications of the ACM*, *55*(5), 10. Retrieved 2021-05-04, from https://mags.acm.org/communications/201205?pg=12#pg12

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: a modern approach* (Third ed.). Upper Saddle River: Prentice Hall.

Yato, T. (2003). *Complexity and completeness of finding another solution and its application to puzzles* (Master's thesis, University of Tokyo, Tokyo, Japan). Retrieved 2021-05-04, from http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.pdf

Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., & Xu, C. (2020, October). Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. *arXiv:2010.12367 [cs, stat]*. Retrieved 2021-03-31, from http://arxiv.org/abs/2010.12367