SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

On Automated Price Model Choice in the Cloud

Marcel Dietl

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

On Automated Price Model Choice in the Cloud

Über die automatisierte Auswahl von Preismodellen in der Cloud

Chair:	Decentralized Information Systems and Data Management
Supervisor:	Prof. Dr. Viktor Leis
Advisor:	M.Sc. Maximilian Kuschewski
Author:	Marcel Dietl
Submission Date:	14.08.2023

Abstract

Public cloud providers offer a wide variety of hardware instances with distinct characteristics, each available in different pricing models. While on-demand is the standard pricing option, there are several alternative options such as reserved instances, savings plans, or spot instances, all of which offer significant discounts and thus opportunities to reduce costs. With the overall goal of cost-optimization, it is important to not only select and deploy the appropriate instance types for specific workloads, but also to efficiently leverage the benefits of different price plans. The model proposed in this thesis aims to employ the best pricing options along with suitable hardware resources for different parts of workloads, ultimately approaching cost-optimality. To do this, the model correlates the specifications of particular pricing options with the characteristics of certain parts of a workload to determine a cost-optimized solution. An evaluation of the model with real-world data traces demonstrates its viability and effectiveness in determining cost-optimized instance configurations and price plans for workloads of various patterns and sizes. The key factors that influence a cost-optimized solution are discussed, and several ways to build on the proposed model are outlined.

Contents

Abstract **iii**

1 Introduction **1**

2 Background and Related Work **4**

2.1 Background . . . 4

2.2 Related Work . . . 5

3 Approach **9**

3.1 Scope and Assumptions . . . 9

3.2 Instance and Pricing Data . . . 10

3.3 Basic Model (M1) . . . 10

3.4 Base and Fluctuating Workloads (M2) . . . 11

3.5 Migration Costs and Interruption Frequencies (M3) . . . 13

3.6 Growth Rate (M4) . . . 16

3.7 Optional Parameters (M5) . . . 16

4 Evaluation and Discussion **19**

4.1 Evaluation . . . 19

4.1.1 Scenario 1 - Large virtual warehouse . . . 19

4.1.2 Scenario 2 - Middle sized virtual warehouse . . . 24

4.1.3 Scenario 3 - Small virtual warehouse . . . 26

4.2 Discussion . . . 28

4.2.1 Base workload pricing options . . . 29

4.2.2 Parallelization fraction . . . 30

4.2.3 Processor architecture . . . 32

4.2.4 Spot pricing . . . 32

4.2.5 Workload growth . . . 34

4.3 Limitations . . . 36

5 Future Work **38**

6 Conclusion **41**

Abbreviations **42**

List of Figures **43**

List of Tables **44**

Bibliography **45**

1 Introduction

Over the past decade, cloud computing has experienced a tremendous growth in popularity. Major public cloud providers like Amazon Web Services (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP) recorded an exponential increase in their revenues [1]–[3]. In 2020, worldwide enterprise spending on cloud infrastructure services surpassed that of on-premise data centers [4]. Cloud computing offers numerous advantages for customers, such as flexibly scaling resources up and down based on current demands, paying only for the resources that are actually used on a per-second basis (pay-as-you-go pricing), as well as high reliability and availability of provided infrastructure. However, despite the diverse benefits of cloud computing, there has been a recent setback in this trend due to the rising costs of cloud services [5].

Public cloud providers offer a wide range of services to their customers. From basic compute infrastructure using Virtual Machines (VMs) (e.g., Amazon's Elastic Compute Cloud (EC2) or Google's Compute Engine), over object storage services such as Amazon S3 or Azure Blob Storage, to cloud Software as a Service (SaaS) data warehouses like Redshift for AWS [6] or Snowflake for AWS, Azure, and GCP [7]. While services such as Google BigQuery or Amazon Athena allow users to run analytical queries and pay only per data access, they are still quite expensive. Previous work shows that by comparing the costs of AWS Athena with the costs of a hypothetical query engine running on AWS EC2, Athena is 50 times more expensive when using on-demand prices for EC2 instances, and even 100 to 200 times more expensive if other pricing options providing discounts are used [8].

To efficiently leverage cloud offerings, it is crucial to take an economic perspective. While managed cloud services such as AWS Athena are easy to use and provide great opportunities for customers, they can be very costly compared to what is theoretically achievable. But even by relying only on computing services like EC2 - on top of which many of the other cloud services are implemented [9], [10] - it is challenging to achieve cost-optimality, which is the minimum monetary cost for executing a particular workload in some cloud hardware landscape [8]. This complexity arises from the fact that each of the public cloud providers offers an extensive variety of VM types, differing for example in the provided hardware resources such as computing power through virtual CPUs (vCPUs) (i.e, CPU cores), network performance (i.e., bandwidth), or memory (i.e., DRAM). Taking EC2 as an example, there are more than 600 different instances available, clustered and optimized for specific use cases [11]. For instance, compute-intensive tasks like high-performance-computing jobs require more vCPUs, while memory-intensive tasks such as big data processing requires more DRAM. On EC2, memory-optimized instances deliver DRAM for costs as low as $0.005 per GiB (x2gd instance), but computing power as expensive as $0.084 per vCPU. Conversely, compute-optimized instances provide computing power for costs as low as $0.034 per vCPU (c6g instance), but DRAM as expensive as $0.017 per GiB. Even though there are also more balanced, general-purpose VMs like the m5 instances, this still highlights the necessity of making trade-offs between hardware resources in practice.

In addition to the large variety of instance types, multiple price plans are available for each instance, offering various discounts that can result in substantial cost savings of up to 90% compared to on-demand prices. To demonstrate the amount of costs that can be saved by aiming for cost-optimization, consider a simplified scenario where a workload is CPU-bound, meaning that it needs to be optimized by computing resources, and runs on any hardware architecture. Further suppose that the customer running this workload previously used a c5 instance (Intel processor) at a cost of $0.043 per vCPU using on-demand prices. Switching to a c6g instance (ARM-based processor) with an on-demand price of $0.034 per vCPU would already result in cost savings of 21%. By further opting for an alternative price plan on this instance would yield additional savings ranging from 27% to 62% depending on the pricing option. This emphasizes the importance of leveraging different instance types and price plans, as it significantly reduces cloud spending.

Therefore, to achieve cost-optimization in the cloud and effectively minimize expenses, for instance in use cases like analytical query processing, it is essential to focus on reducing costs associated with the provisioning of VMs, as they are the underlying service that provides the required computing capacity for such tasks. However, selecting suitable instance configurations from the vast array of instance types and price plans is a challenging and time-consuming task. While best practices and rules-of-thumb exist for choosing a particular instance [12] or pricing option [13] for a specific use case, an automated solution that provides a cost-optimized instance configuration and price plan for a given workload is currently lacking. Therefore, the goal of this thesis is to provide a simple, yet effective model to determine a cost-optimized instance configuration and pricing option available for a particular workload demand. Most workloads running in the cloud do not require a high level of sustained CPU performance [14]. They rather demand continuous, low level CPU performance, and the ability to cover occasional workload spikes. For this purpose, multiple pricing options are leveraged to find and adapt instance configurations to different parts of a workload, ultimately achieving cost-optimization.

The thesis is structured as follows. Section 2 explains relevant background and covers related work to optimizing cloud costs. In Section 3, the proposed model to achieve cost-optimization is gradually derived. It is evaluated and discussed using real-world data traces in Section 4. Section 5 outlines possibilities for future work to build upon the model and Section 6 concludes the thesis.

2 Background and Related Work

2.1 Background

In the following sections, specific instance types and price plans correspond to AWS EC2, as EC2 data was utilized for implementing and evaluating the proposed model. However, equivalents for instance types and price plans also exist on Azure and GCP [15], [16]. A comparison of both is shown in Table 2.1 and Table 2.2. Overall, the fundamental concept of the model is independent from specific instance types or pricing details as the underlying principles of instance selection and pricing optimization remain consistent.

Table 2.1: Selection of instance families on AWS, Azure, and GCP

	General Purpose	**Compute Optimized**	**Memory Optimized**
AWS	M Series (e.g., m6a)	C Series (e.g., c5n)	R Series (e.g., r7g)
Azure	D Series (e.g. D16v3)	F Series (e.g. F16s v2)	M Series (e.g. M32ts)
GCP	N2D Series (e.g. n2d-highcpu-16)	C2D Series (e.g. c2d-highcpu-16)	M2 Series (e.g. m2-megamem-416)

Table 2.2: Cost savings (compared to on-demand prices) of main pricing options on AWS [17], Azure [15], and GCP [18]

	Savings Plans	**Reserved Instances**	**Spot Instances**
AWS	Up to 66%	Up to 72%	Up to 90%
Azure	Up to 65%	Up to 72%	Up to 90%
GCP	28% for 1 year 46% for 3 years	37% for 1 year 57% for 3 years	Between 60% and 91%

Instances on EC2 are organized into clusters optimized for specific use cases, known as instance families. For example, the "c" prefix in instance names signifies that they belong to the compute-optimized instance family [19]. Within each instance family, there are various instance types available (e.g., c5 or c6g), differing in the hardware used to power the instances, such as the type of processors (Intel, AMD, ARM). Each instance type includes one or more instance sizes (e.g., c5.xlarge or c5.2xlarge), enabling customers to scale their resources up or down based on their current requirements. Pricing for an instance type scales linearly with the size of the instance, which means that doubling the resources on one instance type also doubles the associated costs.

The pricing options for these instances include four main categories: On-demand pricing, Reserved Instances (RIs), Savings Plans (SPs), and Spot instances. With on-demand pricing, customers are billed for the computing resources used on a per-second basis. While this option offers maximum flexibility, it is also the most expensive one. All discounts provided by other price plans are compared to the respective on-demand prices. RIs offer discounts based on a committed instance utilization [20]. There are two types of RIs: standard and convertible. Standard RIs provide the highest discounts, up to 72%, by committing to a specific instance type for a one- or three-year term. Convertible RIs offer slightly lower discounts, up to 66%, but provide greater flexibility as they allow changes in instance family or operating system. SPs are very similar to RIs, but are based on committed usage (i.e., hourly spend in $/hour) [21]. There are two types of SPs corresponding to the types of RIs. EC2 Instance SPs offer discounts of up to 72% and require an hourly spend commitment for one or three years within a specific instance family and region. The more flexible Compute SPs offer up to 66% of cost savings and can be applied across instance families, regions, and two other AWS services, Fargate and Lambda. Both RIs and SPs offer three payment options: all upfront, partial upfront, and no upfront. The highest discounts of up to 90% can be obtained through spot instances [22]. Spot instances are unused EC2 capacity available at fluctuating prices determined by supply and demand. While spot instances provide flexibility without any commitment at very high discounts, they can be interrupted and reclaimed by EC2 with short notice when required elsewhere.

2.2 Related Work

Foundation. This thesis is based on the foundational work of Leis and Kuschewski [8], who presented the visionary, long-term goal of creating a cloud-native Online Analytical Processing (OLAP) system that approaches cost-optimality. They observed a growing disparity between the potential capabilities of modern hardware and the actual performance achieved by real-world systems. To address this, their objective is to develop an OLAP system that minimizes costs to the theoretical minimum, which is the underlying hardware expenses. As a starting point, they constructed a model capable of estimating the runtime and cost of a workload on a particular hardware configuration, assuming that all hardware resources (vCPU, DRAM, SSD, Network) can be fully exploited. Using this model, they can predict the most cost-effective instance configurations for various workloads, primarily relying on on-demand prices. However, to further approach cost-optimality on EC2, it is essential to leverage other pricing options available, such as RIs, SPs, or spot instances, as they offer substantial cost savings compared to on-demand prices. This is where the model presented in this thesis ties in.

Instance selection based on workload characteristics. To achieve cost-optimization, it is crucial to utilize instances that align with the specific characteristics of a workload. Wei *et al.* [23] have highlighted that cloud costs are often unnecessarily high due to the mismatch between utilized hardware resources and workload demands. For example, employing an instance with high computing power but insufficient memory for a memory-intensive task results in resource wastage on the one hand, and diminished performance on the other hand. Therefore, minimizing costs entails selecting instances that closely match the workload requirements. To simplify workload-dependent hardware selection, the authors propose a heterogeneous VM allocation algorithm to identify suitable instance types for specific, given workload demands. Maximum utilization of hardware resources provided by particular instances is also an essential part of the proposed model, but is additionally used in combination with the best available pricing options.

Predicting workloads and instance configurations. Several other studies have attempted to effectively predict workloads, rather than taking them as given, and developed algorithms which identify the best instance configurations for them. However, automatically optimizing instance selection while minimizing search costs is a challenging task due to the vast array of potential instances available in the cloud. For instance, CherryPick [24] is a framework that employs performance models for specific instance configurations and employs Bayesian optimization to approach optimal solutions with minimal overhead. In contrast, Hsu *et al.* [25] argue that Bayesian optimization can be fragile due to the complexity of modeling performance and workload relationships. Instead, they propose a method that relies on low-level performance metrics and historical data to effectively and reliably determine the best instance configurations. Another noteworthy finding in this area from the same authors is that a single instance configuration is often near-optimal for multiple workloads simultaneously [26]. As a result, they advocate optimizing multiple workloads collectively to reduce measurement costs while still obtaining satisfactory outcomes. Bilal *et al.* [27] analyzed and evaluated various of such approaches, including the previous ones, and concluded that there is no single best method to determine a cost-optimized instance configuration for a given workload. This highlights the complexity of the challenge to determine a cost-optimized instance configuration for a particular workload, even when pricing options are not yet taken into account. For that matter, the proposed model reduces the complexity in the search for a cost-optimized instance configuration by only considering computing demand for workloads and computing capacity for hardware resources.

Leveraging spot instances. While the aforementioned studies employ sophisticated approaches to find the best instance configurations by analyzing hardware specifications and aligning them with specific workloads, they do not take different pricing options into consideration. However, incorporating these pricing options can be a promising strategy for achieving cost-optimization, especially with the availability of substantial discounts, particularly when utilizing spot instances. Kaulakienė *et al.* [28] developed a framework that leverages spot instances. They estimate the execution time of a workload by executing a subset of the workload on a subset of an instance configuration. Based on optimization parameters like costs, their framework suggests which spot instances to deploy. While the general idea of leveraging pricing options to optimize workload execution for costs is similar here, the proposed model in this thesis goes beyond focusing solely on spot instances. Instead, it aims to strike a cost-optimized balance between parts of a workload that are best covered by committed pricing options like RIs or SPs, and parts of a workload that are fluctuating and are thus best covered by more flexible pricing options such as spot instances. Thereby, the current model seeks to minimize costs while balancing the efficient coverage of workload demands and the utilization of multiple pricing options.

Cloud functions for cost-optimization. In their research, Bian *et al.* [29] aimed for cost-optimization in OLAP systems by employing cloud services different from EC2. They analyzed cloud function services like AWS lambda with regards to their cost-effectiveness when processing query workloads. Their findings revealed that cloud functions tend to be more expensive compared to VMs when applied to continuous workloads due to their limited scalability. However, they suggest a hybrid approach using VMs and cloud functions together. They show that by using a scalable VM cluster for sustained workloads and invoking cloud functions to cope with workload spikes, higher performance per price ratios can be achieved compared to state-of-the-art serverless query engines. Even though they specifically focused on query processing, while the proposed model in this thesis is rather general, both approaches differentiate between continuous and fluctuating parts of workloads. In contrast to employing different cloud functions, the current model focuses on leveraging different pricing options and instance types to minimize the costs for different parts of a workload.

Workload analysis. On public clouds, many general purpose workloads do not require a high level of sustained CPU performance [14]. Instead, workloads are rather characterized by a continuous and modest CPU demand with occasional bursts of higher demand. To handle such different requirements cost-efficiently, various price plans can be leveraged. For steady workloads that mostly remain on the same level, committed pricing options like RIs or SPs are suitable. However, for fluctuating workload spikes occurring irregularly, flexible spot instances without any commitment are more appropriate. Real-world data traces from the data warehousing service Snowflake [30] show how such workload patterns look in practice (see Figure 2.1). In the so called Snowset, statistics on approximately 69 million queries across all virtual warehouses during a two-week period in February/March 2018 were published. An analysis of this dataset reveals that real-world workloads vary significantly in workload size, pattern and type (i.e., read-only, write-only, or read/write queries) from customer to customer [31]. Consequently, a comprehensive model is required that provides cost-optimized instance configurations and price plans for the diverse array of potential workloads. The model proposed in the following section aims to achieve this.

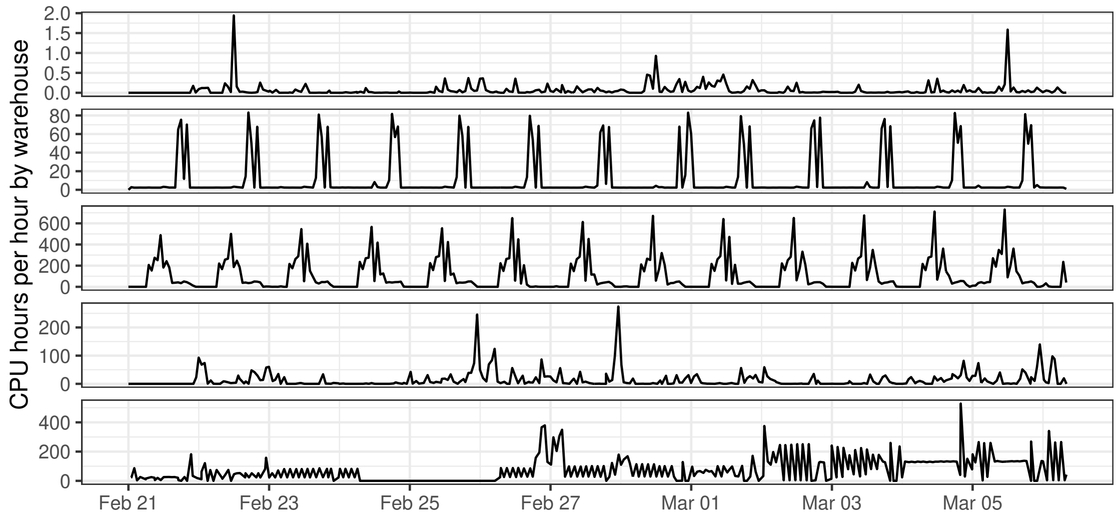


Figure 2.1: Exemplary Snowflake workloads of different virtual warehouses.

3 Approach

This thesis is based on the idea of achieving cost-optimization by leveraging various pricing options available for different parts of a workload. However, determining a particular instance configuration for a specific workload on the one hand, and a suitable price plan for it on the other hand, is a complex task. For example, which instance type is best for a given workload? Is one instance type sufficient for all parts of the workload or are various instances required to achieve cost-optimization? How much of a workload should be covered by committed price plans like RIs or SPs? And how much is worthwhile to be covered by more cost-efficient, but fluctuating spot instances?

The proposed model aims to answer those questions. This section outlines the model's scope and assumptions, explains the utilized data, and describes the incremental steps taken to gradually make the model more complex and realistic.

3.1 Scope and Assumptions

A major challenge when optimizing costs is to decide which hardware resources are important for particular workload types. For instance, high-performance-computing workloads require more computing power, while in-memory databases require more memory. In the proposed model, workloads are measured in CPU hours per hour (CPUh/h), where one CPU hour (CPUh) represents the computing work performed by one processor within one hour. Therefore, to simplify the hardware selection process, an instance is solely characterized by its provisioned vCPUs as it determines the amount of workloads that can be processed within a specific period of time. For example, taking a workload of 8 CPUh/h, an EC2 instance with 8 vCPUs (such as an a1.2xlarge) can process this workload within one hour, assuming that the computing capacity can be fully exploited. Consequently, other hardware resources such as network performance or memory are neglected in this model. In cases where workloads exceed the capacity of a single machine or the usage of multiple machines offers lower costs, it is assumed that scaling out is only possible in homogeneous clusters, meaning instances of the same instance type and size.

3.2 Instance and Pricing Data

The model utilizes original EC2 instance and pricing data obtained from AWS via the Python AWS SDK called boto3. The data was retrieved on June 18th, 2023, from the region us-east-1 (North Virginia) for Linux instances with shared tenancy. The dataset includes pricing information about on-demand prices, RIs, SPs, and spot instances as well as spot interruption frequencies. Pricing information for RIs comprises both standard and convertible RIs in both commitment terms (i.e., one or three years) and all payment options (i.e., no upfront, partial upfront and all upfront). As RI prices can be specific to an Availability Zone (AZ), they were averaged over three AZs in region us-east-1 (i.e., us-east-1a, us-east-1b, and us-east-1c). Prices for instance configurations on the RI marketplace are not included. Pricing information for SPs include both EC2 Instance and Compute SPs, also in both commitment terms and all payment options. Spot instance prices were obtained for three AZs (i.e., us-east-1a, us-east-1b, and us-east-1c) over the last three months. Both the average price and the minimum price for each instance were calculated. In general, burstable instances were excluded from the model due to their specific functionality.

3.3 Basic Model (M1)

In the initial version of the model, a workload is characterized only by one constant CPUh/h value. It is assumed that all workloads occurring are critical, thus they must be covered by uninterruptible instances, disallowing the usage of spot instances.

The goal is to determine a cost-optimized instance configuration with the best pricing option for a given workload. Taking a hypothetical workload demand of 8 CPUh/h, to model searches for instances that provide 8 vCPUs, thus being capable of executing the workload by fully exploting all CPU cores. It compares the possible pricing options (on-demand, RI, SP) for all potential instances and determines an a1.2xlarge instance to be best using an EC2 Instance SP with a three year commitment and an all upfront payment. This incurs hourly costs of \$0.0767.

If the workload is now doubled to 16 CPUh/h, the question arises whether it is more economical to scale up or to scale out, which in this case means either scaling up to one instance with 16 vCPUs such as an a1.metal (\$0.153), or scaling out to two instances with 8 vCPUs each, such as the previously used a1.2xlarge ($2 * \$0.0767 = \$0.153$). As instance prices on EC2 scale linearly with the size of the instance within one instance family (e.g. a1), the two options do not differ in terms of costs.

However, in reality using distributed systems, parallelization across different machines using multiple processors does not scale perfectly. According to Amdahl's Law [32], the theoretical speedup of a program execution is limited by the fraction of the part of the program that cannot be parallelized. Take the example of a program that needs 20 hours to complete using a single thread, and 1 hour of that cannot be parallelized. This means that only 95% of the program can be executed in parallel and therefore implies that the program at least needs 1 hour to complete, even with an infinite number of machines and processors. Thus, the upper bound for a theoretical speedup in this scenario would be 20. Amdahl formalized this with the following equation:

$$s = \frac{1}{(1-p) + \frac{p}{n}} \tag{3.1}$$

where s is the speedup achieved by parallelizing the program, p is the proportion of the program that can be parallelized and n is the number of machines used for parallel execution. By solving for n the following formula is obtained:

$$n = \frac{p}{\frac{1}{s} - (1-p)} \tag{3.2}$$

which can be used to determine the number of machines necessary to achieve a desired speedup.

To illustrate the consequences for the model, assume a hypothetical hourly workload of 16 CPUh/h as in the example from above. Again, the question arises whether it is more cost-efficient to use a single larger instance (such as an a1.metal) or multiple smaller instances in a homogeneous cluster (such as two a1.2xlarge). Applying the model using Amdahl's formula (see 3.2) with a desired speedup of $s = 2$ and a parallelization fraction of $p = 0.95$, the amount of actually required instances of $n = 3$ is obtained. This means that the total hourly costs of using two a1.2xlarge instances increases to $3 * 0.0767\$ = 0.230\$$. Thus, the cost-optimized instance configuration changed from an indifferent scenario between one a1.metal and two a1.2xlarge instances, to an a1.metal being the strictly better option with costs of 0.153$ per hour.

3.4 Base and Fluctuating Workloads (M2)

In M1, it is assumed that a workload measured in CPUh/h remains constant over time and can be represented by one fixed parameter. However, this is rather unrealistic in practice. As the previously mentioned analysis of the Snowset [30] shows, workloads vary significantly among different customers [31]. For instance, looking at the data of one particular day, some customers had periodically fluctuating workloads every hour, while others had a few highly demanding workloads once or twice a day, while still others were consistently active but with varying levels of noise in their workloads. Applying M1 to such real-world workload demands would result in a substantial waste of resources. As Figure 3.1a shows for an illustrative daily workload demand, M1 would suggest acquiring an instance configuration with a committed price plan that is capable of executing all workloads. As the workload peak is well above most of the other workloads, a large amount of computing capacity is wasted during hours of lower demand. Obviously, optimized resource utilization and thus cost-efficiency cannot be achieved in this way.

To address this issue, the constant parameter already used in M1 now represents a fraction of the overall workload called base workload and is characterized by the fact that it is critical and must be consistently covered at all times. This includes jobs that run continuously and therefore require constant computing resources. In addition, the model is extended by a second parameter called fluctuating workload that covers the remaining fluctuations and spikes in workloads not covered by the base workload. As described above, fluctuations can vary greatly from hour to hour, often resulting in 24 different fluctuating workload demands throughout a day. Thus, the idea of M2 is that the base workload covers a specific amount of CPU demand, while spikes exceeding the base workload are covered by the fluctuating workload.

As in M1, the base workload is covered by the cost-optimized instance configuration using the best pricing option out of on-demand, RI, and SP options. Those are well-suited for base workloads, as once one has committed to a specific configuration, instances cannot be interrupted. In contrast, even though spot instances often provide even greater discounts, they can be interrupted and reclaimed by EC2 at any time, potentially interrupting currently running jobs. As fluctuating workloads are defined by M2 to not require continuous execution like base workloads, the possibility of interruptions can be tolerated. Moreover, due to the often varying hardware requirements of fluctuating workloads, spot instances with their flexibility and no timely commitment are an important part in finding a cost-optimized solution. By incorporating base and fluctuating workloads, M2 aims for better workload coverage, ultimately leading to improved resource utilization and cost-optimization.

To illustrate how M2 works, assume a simplified workload distribution shown with the dotted line in Figure 3.1b. Except from one workload spike reaching 24 CPUh/h and a consecutive spike reaching 32 CPUh/h, all other workloads fluctuate between 6 and 8 CPUh/h. Therefore, by applying the logic from M2, it is beneficial to split the workloads into two parts. One fixed base workload that covers all relatively constant workloads. And one fluctuating workload part, which in this case comprises two values (i.e., the workload spikes of 24 CPUh/h and 32 CPUh/h). Using M2 with those considerations, one a1.2xlarge instance ($0.0767 per hour) is suggested to cover a base workload of 8 CPUh/h by using an EC2 Instance SP with a three year commitment and an all upfront payment. Additionally, one c5.4xlarge instance is acquired as a spot instance ($0.3686 per hour) to cover the first workload spike, and one r5.8xlarge spot instance ($0.5715 per hour) is purchased to cover the second spike.

Note how the spot price for the c5.4xlarge is actually more expensive compared to its cheapest SP option available for $0.2480 per hour. This is due to the fact that on average, this instance had a high demand over the last three months, resulting in spot prices that offer lower discounts compared to fixed RI or SP options. Nevertheless, it is still the most cost-effective spot instance available for this particular workload spike. Overall, choosing this combination of base and fluctuating workloads results not only in better resource utilization, but also in significantly lower costs compared to the solution of M1. Extrapolating the hourly costs of M1 shown in Figure 3.1a to the day results in total costs of $9.2040. In contrast, M2 lowers the daily costs to a total of $2.7809, achieving cost savings of nearly 70%.

3.5 Migration Costs and Interruption Frequencies (M3)

In theory, it might be reasonable to switch a spot instance configuration as often as hourly to optimally cover fluctuating workloads. However in practice, migrating from one spot instance configuration to another is not free, as - for example - the entire instance memory of a previously used instance needs to be copied, as it is otherwise deleted as soon as an instance is terminated [33]. To perform such clean up tasks, some CPU time is required and thus costs are incurred. As with workload demands in general, migration costs also vary from use case to use case.

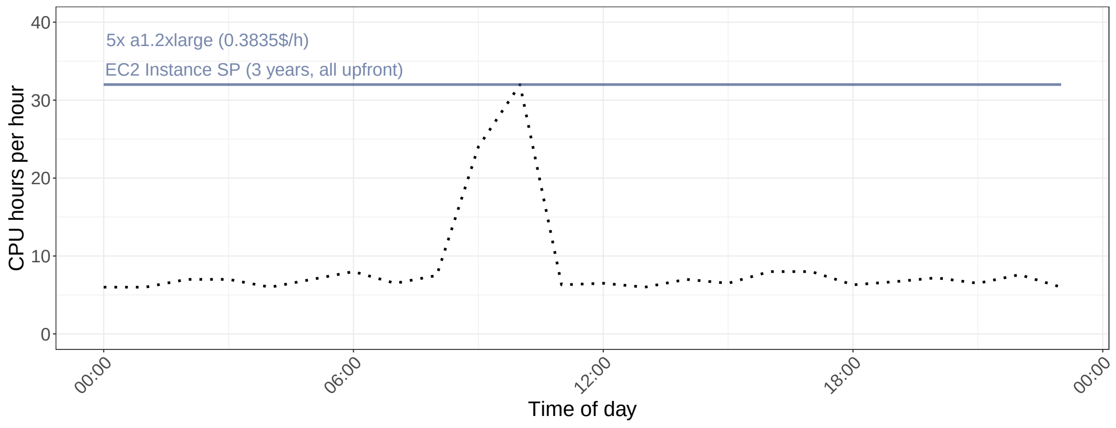
To take this into account, M3 is extended by a third parameter called migration costs. These costs are also measured in CPUh/h and allow including an expense for instance migrations. On the one hand, such migrations may be motivated by a more cost-efficient instance configuration available for new fluctuating workloads, or on the other hand forced by EC2 due to a reclaim of the currently used instance configuration. In both cases migration costs apply.

As a forced interruption by EC2 can happen at any time on any instance configuration, M3 directly incorporates the costs associated with such an event in the spot prices used for each instance. So instead of using the original spot prices obtained from EC2, M3 adjusts them ($s_{adjusted}$) by increasing the original spot price ($s_{original}$) by the amount of the computing time required for a migration (m) using all vCPUs (c) of an instance multiplied by the probability of an interruption (p).

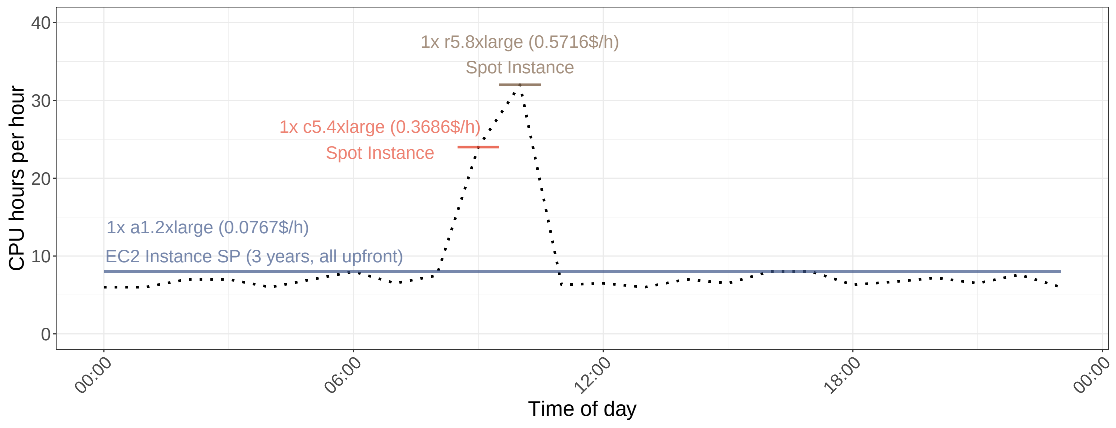
$$s_{adjusted} = s_{original} + (s_{original} * p * \frac{m}{c}) \tag{3.3}$$

As the average monthly interruption frequency for all instances is below 5% [34], the additional costs caused by a forced interruption usually remain extremely marginal, even if the migration costs are relatively high compared to the overall workload demand. However, in addition to the always present monetary costs of a potential reclaim, the actual migration has to be performed in both voluntary and involuntary scenarios. To model the incurring costs, it is assumed that migration efforts are performed on the respective previously used spot instance. If there is no previously used instance or if it is scaled out (or in) on the same instance type, no migration costs incur. As the workload of a migration is specified as a constant, it makes no difference on which instance size the migration is executed.

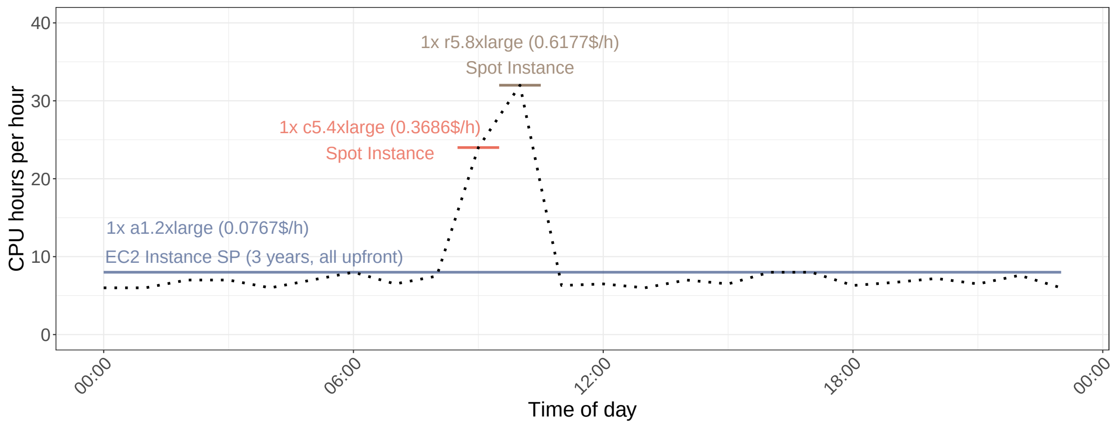
To illustrate the calculation of migration costs in M3, Figure 3.1c depicts that the recommended instance configuration for each hour has not changed compared to M2. However, applying M3 with exemplary migration costs of 2 CPUh/h, a different spot price for the r5.8xlarge instance can be observed. The model added an additional monetary migration cost of $0.0461 to the already adapted spot price of the r5.8xlarge instance. As migration efforts are performed on the previous c5.4xlarge instance, the migration costs of 2 CPUh/h are calculated using its respective hardware resources and are added to the costs for r5.8xlarge afterwards. To decide whether it is more cost-efficient to migrate to a new instance or stay at the previously used instance, M3 compares the newly calculated spot price of the potential new spot instance (here one r5.8xlarge) with the price of the instance used in the hour before (here one c5.4xlarge). As in this case, one c5.4xlarge instance would be insufficient to handle 24 CPUh/h on top of the base workload, the adapted price for an r5.8xlarge is compared with the price of a two c5.4xlarge instance cluster, as scaling out on the same instance type incurs no migration costs in M3. In this scenario, a r5.8xlarge is still the more economical option despite the added migration costs.



(a) Results of model M1



(b) Results of model M2



(c) Results of model M3

Figure 3.1: Instance configurations and price plans with hourly costs according to model M1-M3 covering an exemplary workload demand (dotted line).

3.6 Growth Rate (M4)

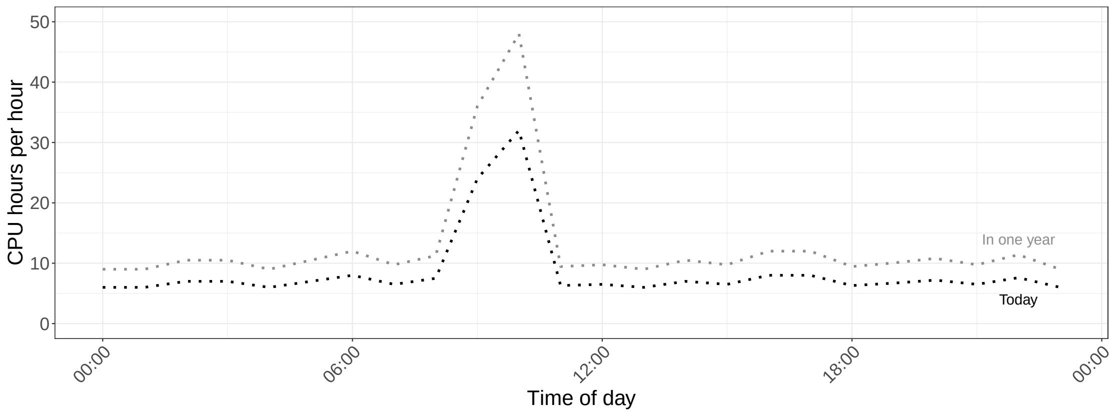
Even though fluctuating workloads might be changing from hour to hour, the average workload demand in the previous versions of the model (M1-M3) remains constant. In practice, companies mostly expect their business to grow by a certain rate in a certain period of time. Correspondingly, their computing workloads in the cloud also grow. For instance, more entries in a database lead to increased query times, and therefore increased costs running these queries. To take that into consideration, M4 is extended by a growth rate expected for a particular period of time.

A simple approach to cover the additional demands resulting from a growth in workloads is to employ more spot instances, as they can be flexibly adjusted to the new workload demands. However, this can significantly increase the overall costs compared to a potentially cost-optimized solution, in which the base workload is also adjusted.

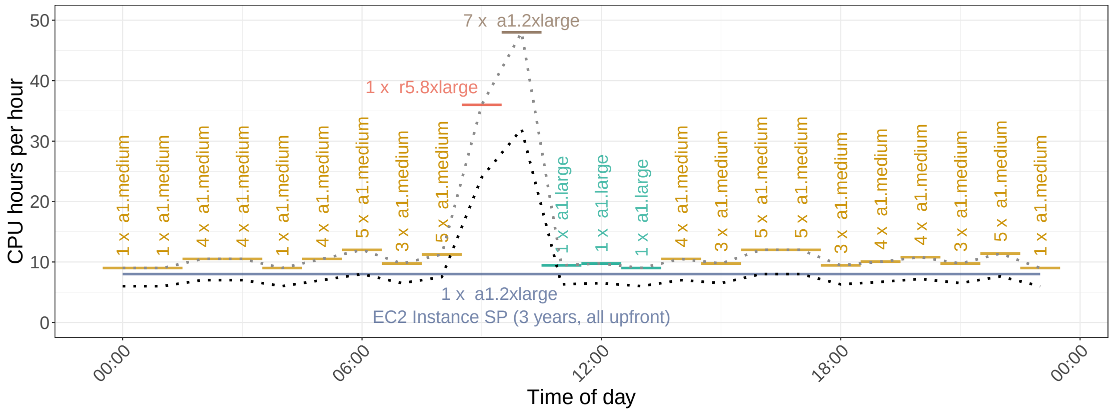
To illustrate this, Figure 3.2a displays the same workload demand as in Figure 3.1, with an additional workload demand (dotted grey line) which results from an exemplary expected growth of 50% within a one year period. As it can be observed in Figure 3.2b, keeping the base workload on the same level requires many additional spot instances to cover the workload growth. In total, such an approach costs $5.62 per day in this example. In contrast, optimizing the workload coverage by increasing the base workload to its original value plus growth (i.e., $8CPUh/h * 1.5 = 12CPUh/h$) incurs daily costs of $5.29 (see Figure 3.2c). Therefore, nearly 6% of costs can be saved by increasing the volume of the utilized SP pricing option to be able to use an a1.metal instance in order to cover all steady workloads. Compared to the solution shown in Figure 3.2c, costs can be reduced even further by exploiting all resources on the a1.metal instance, i.e., by increasing the base workload to 16 CPUh/h. Thereby, daily costs decrease by more than 14% to $4.81 compared to the option of covering growth only with spot instances.

3.7 Optional Parameters (M5)

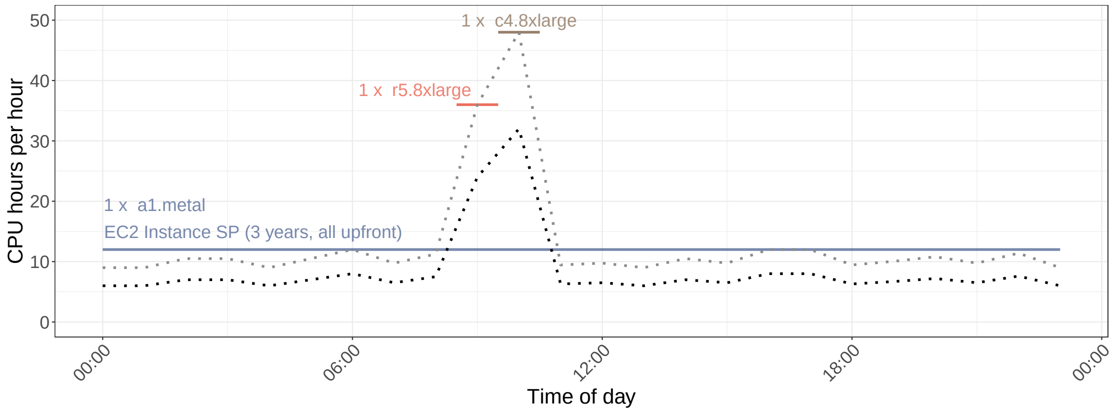
Naturally, EC2 provides the largest discounts for pricing options with low flexibility, a high commitment and an all upfront payment. To account for the fact that the previous models would always select one of those price plans for the base workload, even though it might not be implementable for certain customers, M5 is extended by multiple optional parameters. Thus, the number of potential pricing options can be limited in several dimensions.



(a) Current workload and workload in one year assuming a 50% growth rate



(b) Option 1: Keeping base workload constant and cover workload growth with spot instances



(c) Option 2: Adjusting base workload according to the expected workload growth

Figure 3.2: Illustration for the results of model M4 with regards to workload growth.

First, the general price plan can be selected from on-demand options, RIs, and SPs. Second, the type of price plan can be further specified, meaning that Standard and Convertible RIs as well as Compute and EC2 Instance SPs can be differentiated. Last, duration and payment options can be limited by restricting the model to one or three years of commitment and to a no upfront, partial upfront or all upfront payment. Thus, using the additional parameters, the search space for the pricing options can be limited and therefore the relevance of the model output increased. By default, when no optional parameters are set, the model assumes that each pricing option is possible when searching for a cost-optimized solution. Moreover, the processor architecture of instances can be restricted to either ARM-based or x86 architectures (default is both). Finally, the default parallelization fraction of 95% can be overridden if higher (or lower) parallelization rates can be achieved.
Table 3.1 summarizes all required and optional parameters of the model. Required parameters must always be specified for the model to function, thus having no default values. When none of the optional parameters are specifically set, all displayed options (symbolized by the logical "or" operator) are considered in the search for a cost-optimized solution.

Table 3.1: Model Parameters

Required Parameters	**Default Value**
Base workload (value in CPUh/h)	None
Fluctuating workload (array of values in CPUh/h)	None
Migration Costs (value in CPUh/h)	None
Optional Parameters	**Default Value**
Price Plan	On-demand | RI | SP
Price Plan Type	Standard RI | Convertible RI | Compute SP | EC2 Instance SP
Commitment Term (in years)	1 | 3
Payment Option	No- | Partial- | All-Upfront
Processor architecture	ARM | x86
Parallelization Fraction	0.95

4 Evaluation and Discussion

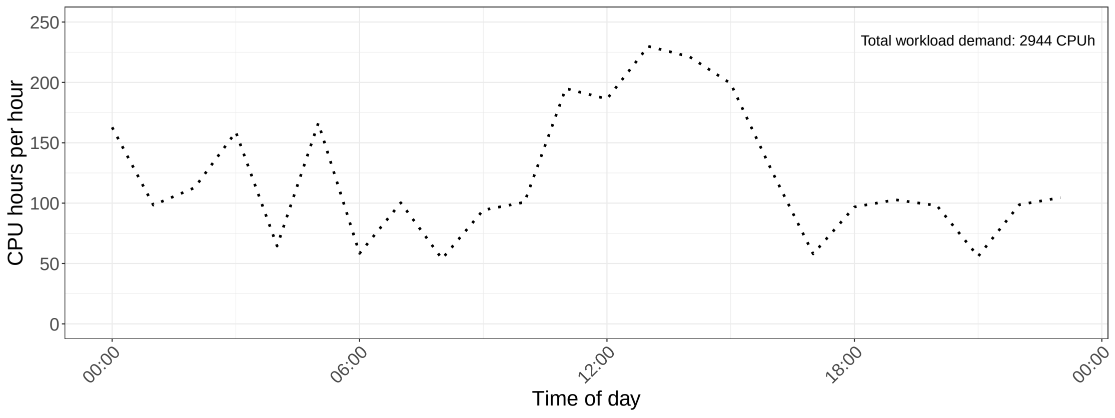
4.1 Evaluation

To assess the effectiveness and practicality of the proposed model when applied to real-world workload distributions, an evaluation is performed using a subset of three different virtual warehouses from the Snowset [30]. This evaluation specifically focuses on the total daily workloads on a particular day (Monday, February 22, 2018), measured in CPU hours. The selected virtual warehouses include workloads with different sizes (large, medium, and small) and patterns, providing a comprehensive evaluation. Initially, the model determines cost-optimized instance configurations and price plans for each virtual warehouse using M3, thus without setting any optional parameters or considering growth rates. Subsequently, the impact of incorporating M4 and M5 on the cost-optimized solution is investigated. Overall, this shows the model's viability in handling diverse and realistic workload scenarios.

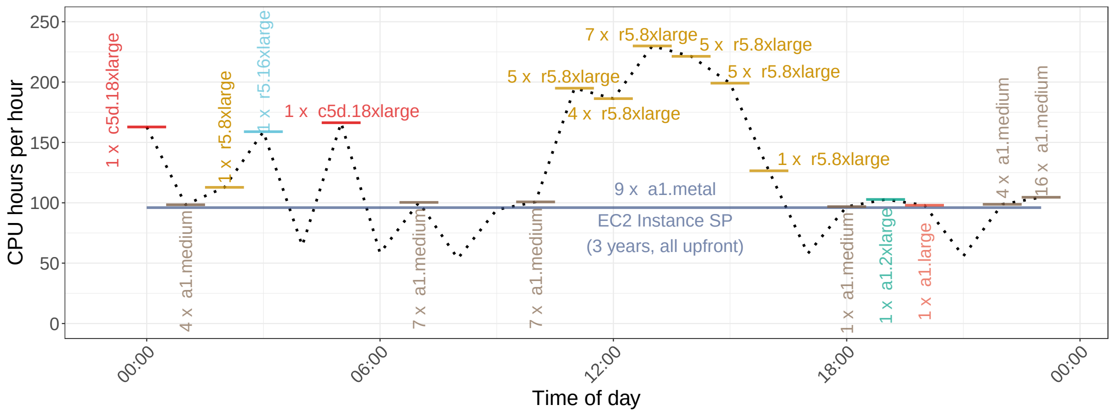
4.1.1 Scenario 1 - Large virtual warehouse

The first virtual warehouse (VH-1) under evaluation ranks as the fourth largest virtual warehouse in terms of CPUh measured by Snowflake on this day. It produced a total daily workload demand of 2944 CPUh distributed across 75066 queries. To apply the model to the workloads of VH-1, single CPUh demands are aggregated and summed for each hour of the day, resulting in 24 different values. Looking at Figure 4.1a displaying the distribution of the workloads, one can observe that VH-1 was active throughout the entire day with substantial, frequently occurring fluctuations.

To determine a set of cost-optimized instance configurations and pricing options for executing those workloads, the proposed model (M3) is applied. With the flexibility to accommodate different workload inputs, various options are conceivable to cover the demand. However, from an economic perspective, it is crucial to identify the solution that provides the highest cost-efficiency. In the subsequent evaluation, this optimized solution is gradually converged to demonstrate the model's effectiveness. In principle, there are two simple variations to accommodate the workload, with the cost-optimized solution being a particular combination of both.



(a) VH-1 - Workload demand



(b) VH-1 - Cost-optimized instance configurations and price plans

Figure 4.1: Workload demand of VH-1 (22-02-2018) as well as instance configurations and price plans for the cost-optimized approach.

Base workload only. The first variation closely resembles an on-premise setup where fixed server capacity must be sufficient for all workload demands. In the context of the model, this means that only a base workload is used to cover all demands, without leveraging any spot instances. As committed pricing options are not flexible, the utilized instance configuration must be capable of accommodating even the highest workload spike. While this obviously also provides adequate processing power for all other occurring workloads, it yields significant waste of computing resources - and thus money - during periods when the workload is less demanding than the maximum. Applied to the workload distribution of VH-1, a fixed base workload of approximately

230 CPUh/h is required. With no fluctuating workloads left, the model determines five c6g.16xlarge RIs with a three-year commitment and an all upfront payment to be best. This results in hourly costs of \$4.09 and daily costs of \$98.16.

Fluctuating workload only. The second variation pursues a contrary approach as it aims to cover all workloads with spot instances only. This allows for a flexible adjustment of the computing resources to the current demand. However, due to the heavy workloads of VH-1, potential migration costs as well as the necessity to deploy large spot instances (or instance clusters) must not be neglected. Therefore, the daily costs still amount to \$85.19 when considering exemplary migration costs of 2 CPUh/h, leading to cost savings of only 13% compared to a base workload-only approach.

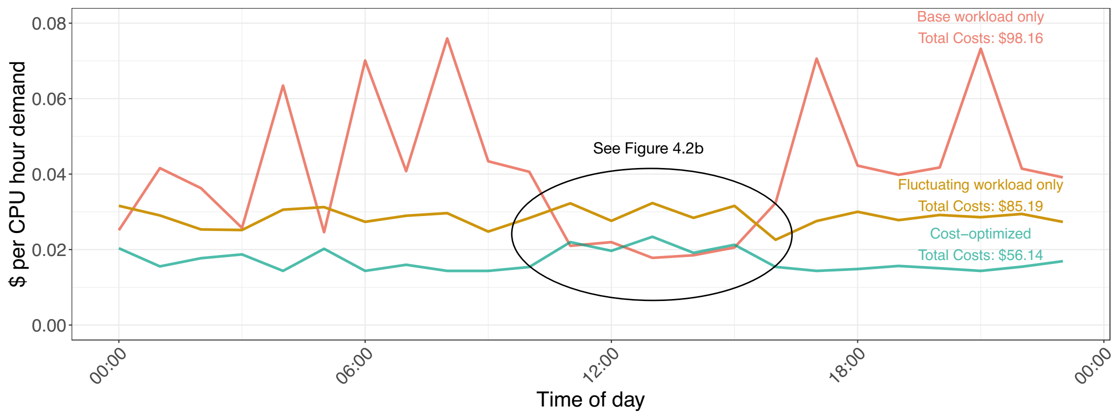
Cost-optimization. While the model still provides the best possible instance configurations and price plans for the one-sided inputs used above, its fundamental idea and objective to achieve cost-optimization is to combine and leverage different pricing options for both base and fluctuating workloads. Thereby, the model seeks to strike a balance between the rigidity of the base workload-only strategy and the volatility of relying solely on spot instances. However, determining the ideal ratio between these workloads can be challenging. The base workload must be set in such a way that the right amount of workloads are covered, while at the same time ensuring the best possible utilization and minimum waste of resources. In addition, there must be sufficient opportunities to leverage spot instances for the remaining workload spikes.

In the workload scenario of VH-1, which experiences a continuously high workload demand with moderate spikes rather than occasional extreme spikes, an initial approach to approximate the ideal ratio between base and fluctuating workloads is to calculate the mean of all hourly workloads throughout the day and set it as the base workload value. Applying the model with this ($\approx$123 CPUh/h base workload, remaining with fluctuating workload) results in daily costs of \$60.49. Thereby, cost savings of 29% compared to the fluctuating workload-only approach and even 38% compared to the base workload-only approach are obtained.
To further optimize costs, the functionality of the model needs to be revisited shortly. Since the relevant metric of an instance is its provisioned vCPUs, maximum utilization on the available hardware resources is essential for cost-optimization. For the previous base workload of 123 CPUh/h, a c6a.32xlarge instance is utilized in a three year, all upfront SP. As this instance provides 128 vCPUs, the costs can be further reduced by increasing the base workload to 128 CPUh/h, fully exploiting its computing capacity. Thereby, the total daily costs decrease from \$60.49 to \$58.74.

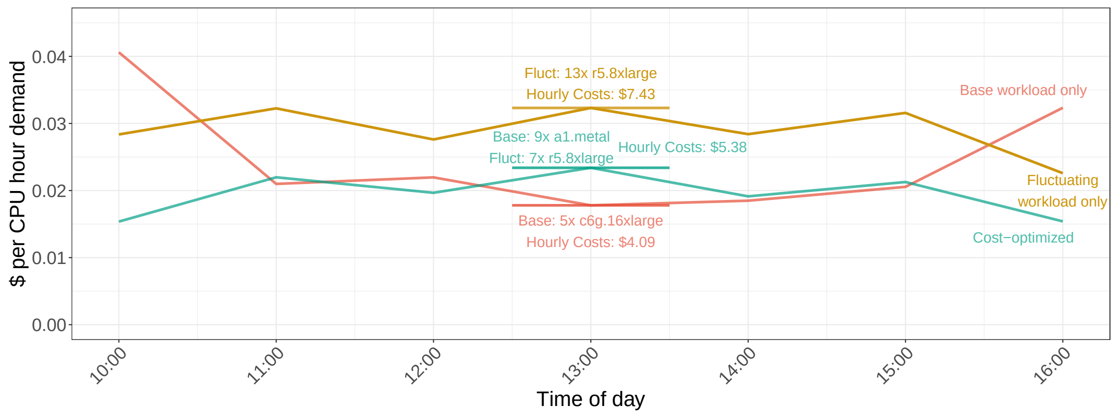
However, to verify whether this already is the optimized ratio between base and fluctuating workloads, other base workloads must be tested in the model. As the minimum hourly workload demand of VH-1 is approximately 54 CPUh/h, it is reasonable to apply the model with values of 196, 96, and 64 CPUh/h, as they all match the provisioned vCPUs of available instances. Respectively, daily costs of \$104.05, \$56.14, and \$59.89 are obtained. Thus, the cost-optimized workload ratio for VH-1 is covering 96 CPUh/h with a committed pricing option and the remaining workload spikes with spot instances. The exact instance configurations are displayed in Figure 4.1b. With that, 34% are saved compared to solely relying on spot instances, and even 43% compared to only using a committed price plan.

Analysis of cost structures. Figure 4.2a illustrates the costs in dollar incurred per hourly workload demand using the respective approaches from above. One can observe that while the costs per hourly workload demand for the fluctuating workload-only and cost-optimized approach are relatively low and stable, those for the base workload-only approach are varying significantly. This can be explained by the fact that the instance configuration and pricing option used in the base workload-only approach is not adjusted to the hourly changing demands as in the other approaches, but rather only selected to accommodate the highest workload spike.
However, it is noteworthy that even though the cost-optimized solution yields overall cost savings of 43% compared to the base workload-only approach, it is still significantly more expensive during the hour of the workload peak. This period is circled in Figure 4.2a and enlarged in Figure 4.2b. The latter also shows the instance configurations and total costs for the time period with the highest workload demand. In order to understand why this occurs, the available pricing options with their respective discounts need to be analyzed.
Table 4.1 depicts five instances with their respective hourly costs and savings for both RI/SP and spot options. While the cost savings provided by the committed pricing options are on a constantly high level of over 60% for all instances, spot discounts vary greatly from below 5% to more than 70%. As the maximum hourly demand of VH-1 requires substantial computing capacity, larger instances (or instance clusters) are required. For the base workload-only approach, the model determines a cluster of five c6g.16xlarge instances with a committed pricing option to be best, amounting in hourly costs of $5 * \$0.82 \approx \$4.09$. For the cost-optimized approach, even though 96 CPUh/h are covered with a SP, there is still significant workload demand to be covered by spot instances. However, as Table 4.1 shows, spot prices are often close to on-demand prices for larger instances. Thus, the best option that remains is deploying a larger cluster of smaller instances (such as the r5.8xlarge) to cover the remaining workload spikes, which also incur increased costs due to Amdahl's law. When using only spot instances,

even larger clusters are required resulting in even higher costs for the workload peak. Nevertheless, it is important to point out that this effect occurs only during time periods where demands are close to the maximum. While the base workload-only approach also costs $4.09 during every other hour of the day, the cost-optimized solution adapts to the respective hourly workload demands, resulting in significantly enhanced overall cost-efficiency.



(a) VH-1 - Costs per hourly CPUh demand for different model approaches



(b) VH-1 - Costs per hourly CPUh demand as well as comparison of instance configurations and absolute costs during the hour with maximum workload demand

Figure 4.2: Costs per hourly CPUh demand of three different model approaches for VH-1.

Table 4.1: Hourly cost and savings comparison (rounded) of selected instances

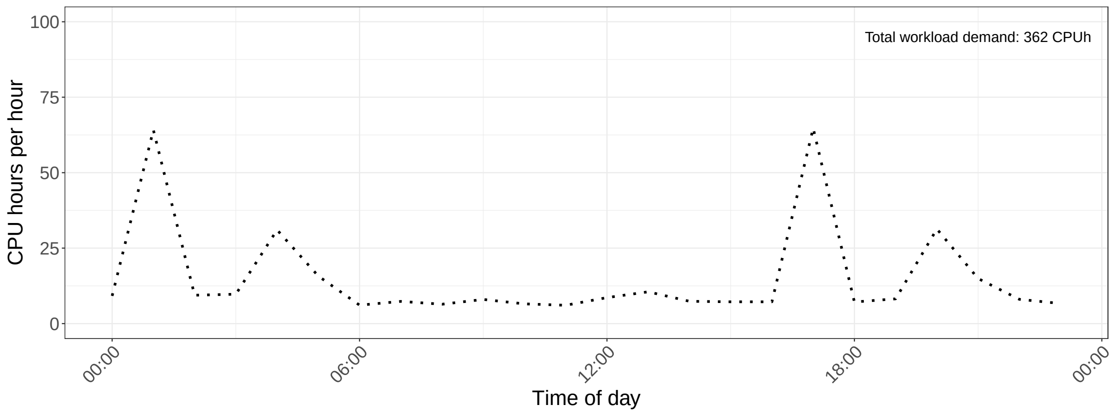
Instance	On-demand price	RI/SP price	Savings	Spot price	Savings
a1.metal	$0.41	$0.15	63.41%	$0.39	4.88%
r5.8xlarge	$2.02	$0.76	62.38%	$0.57	71.78%
c6g.16xlarge	$2.18	$0.82	62.39%	$1.85	15.14%
m6a.32xlarge	$5.53	$2.18	60.58%	$4.63	16.27%
c6a.48xlarge	$7.34	$2.81	61.72%	$7.02	4.36%

4.1.2 Scenario 2 - Middle sized virtual warehouse

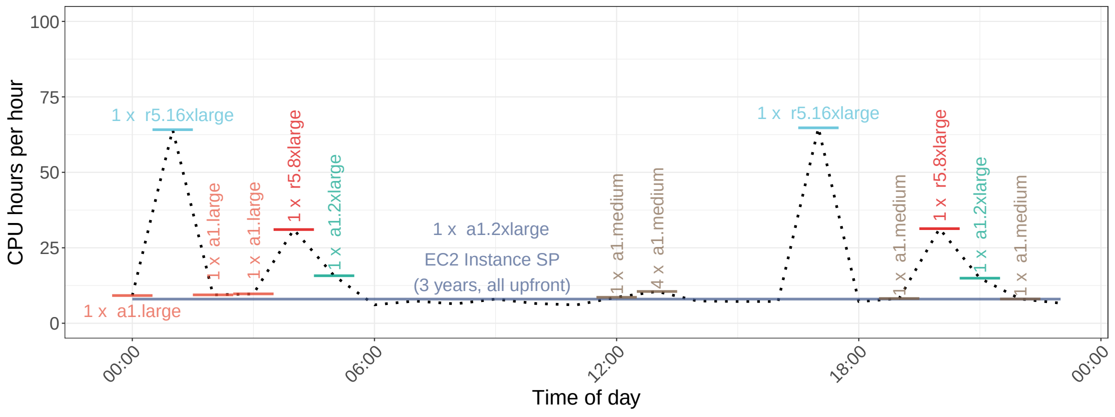
To analyze the model's viability with diverse workloads, it is applied to a second virtual warehouse (VH-2) characterized by a moderate workload demand consisting of 362 CPUh per day distributed across 18,897 queries. Looking at Figure 4.3a it can be observed that the workload demand remains relatively low and stable for the majority of the day, with two distinct spikes occurring twice a day. Similar to VH-1, it is reasonable to assume that a cost-optimized solution for VH-2 involves covering the steady workloads with RIs or SPs and leveraging spot instances for efficiently handling the workload spikes. To put the cost-optimized solution for VH-2 into proportion, the model is also applied with only a base workload and only a fluctuating workload.

Base workload only. Given that the highest workload spikes significantly exceed the majority of the other workloads, which mostly remain continuously low, using only a committed pricing option is far from the most economic solution as it results in a substantial waste of computing resources, even more than in VH-1. By applying the model (M3) with a constant base workload of approximately 65 CPUh/h, the total daily costs reach $25.70, using a cluster of seven a1.metal instances within an EC2 Instance SP with a commitment of three years and an all upfront payment.

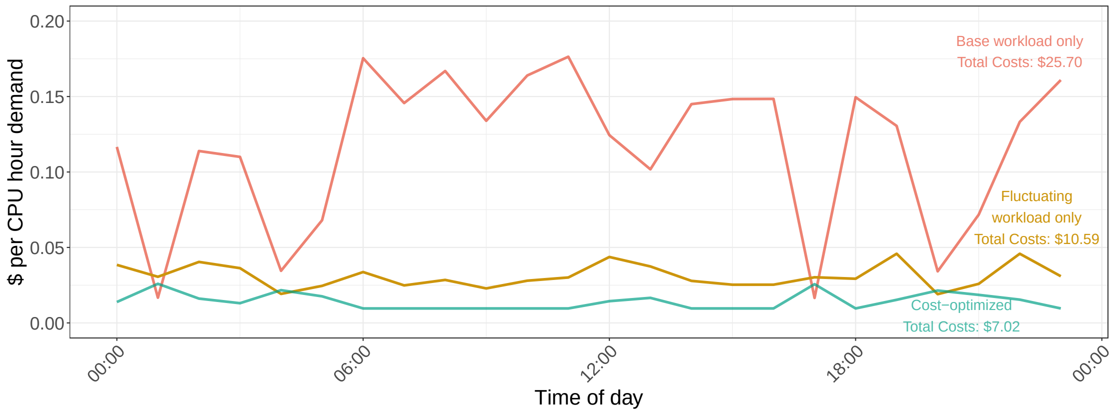
Fluctuating workload only. In contrast, opting for a strategy that exclusively relies on spot instances to achieve a better coverage of the workload demand yields more favorable results. By considering migration costs of 1 CPUh/h, the total daily costs decrease to $10.59. This approach proves to be 59% more cost-efficient compared to relying solely on a base workload as chosen instances are adjusted to the required workloads of each hour, thus wasting less computing resources.



(a) VH-2 - Workload demand



(b) VH-2 - Cost-optimized instance configurations and price plans



(c) VH-2 - Costs per hourly CPUh demand for different model approaches

Figure 4.3: Workload demand of VH-2 (22-02-2018), instance configurations and price plans for the cost-optimized approach, and relative cost distribution.

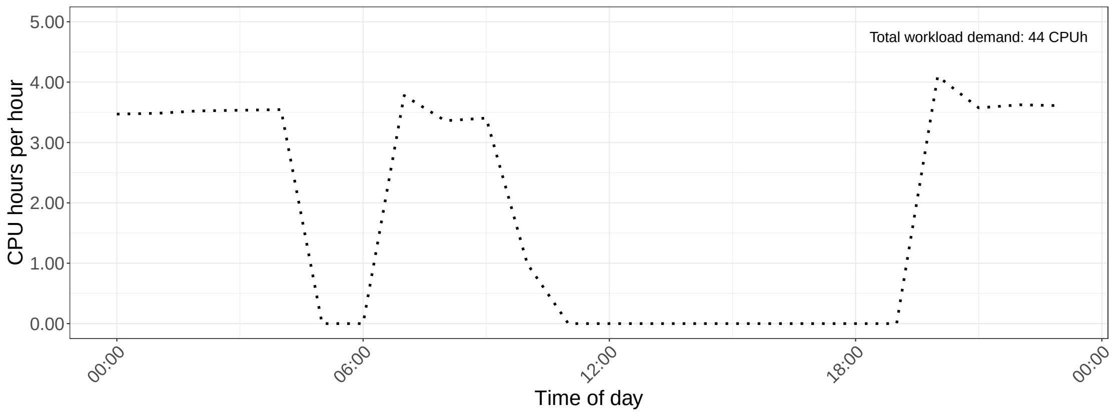
Cost-optimization. By analyzing the workload distribution of VH-2, it is reasonable to estimate the base workload to either 16 or 8 CPUh/h, as both cover most of the low and stable workloads. In fact the cost-optimized ratio between base and fluctuating workloads in this scenario is covering 8 CPUh/h with an EC2 Instance SP and the remaining workloads with spot instances. This is more cost-effective than 16 CPUh/h as a base workload because it provides more opportunities for small a1 instances, which have the best price per performance ratio on EC2. With the instance configurations displayed in Figure 4.3b, the daily costs can be reduced to \$7.02, saving 34% compared to solely relying on spot instances and even 73% to only using a committed pricing option.

Figure 4.3c depicts the costs per hourly workload demand for all approaches explained above. Similar to VH-1, costs per demand for the base workload-only approach vary significantly, while those for the fluctuating workload-only and cost-optimized approach are relatively stable. As there are two extreme spikes in the workloads of VH-2, there are also two time periods during which the base workload-only approach is slightly cheaper than the cost-optimized solution. During these hours, the purchased computing resources in the base workload-only approach are effectively utilized. However, as the workload demand during the rest of the day is significantly lower, resource wastage is even higher as in VH-1. This is also the reason why the cost-optimized solution provides over 70% of cost savings by better adapting to the hourly demands.

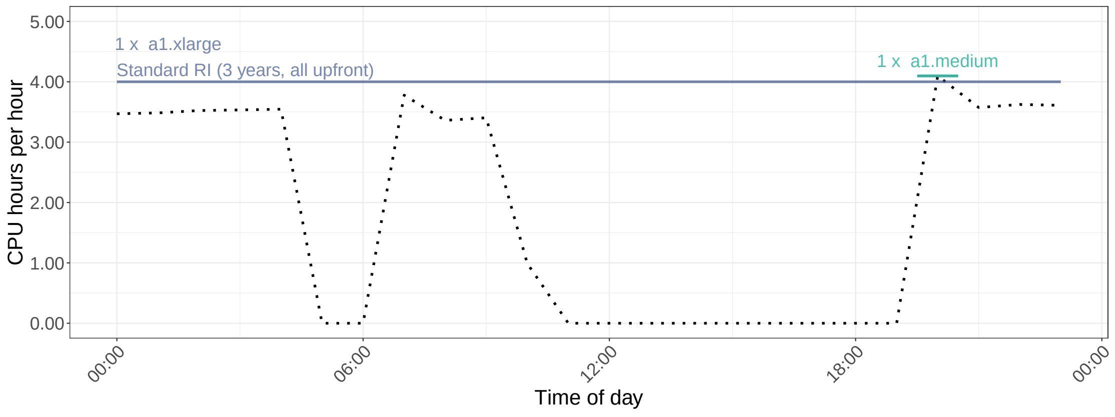
4.1.3 Scenario 3 - Small virtual warehouse

The third and smallest virtual warehouse (VH-3) evaluated with the model had a daily workload demand of 44 CPUh distributed over 279 queries. Besides that VH-3 has a significantly lower workload demand compared to VH-1 and VH-2, it can also be observed that there are time periods during the day where no queries are run and thus no workloads appear. When workload spikes occur, they mostly reach similar amounts of CPUh/h (see Figure 4.4a). As the overall hourly workloads of VH-3 are fairly low, the total costs per day do not differ greatly between different options in absolute numbers. Nevertheless, a cost-optimized solution still exists.

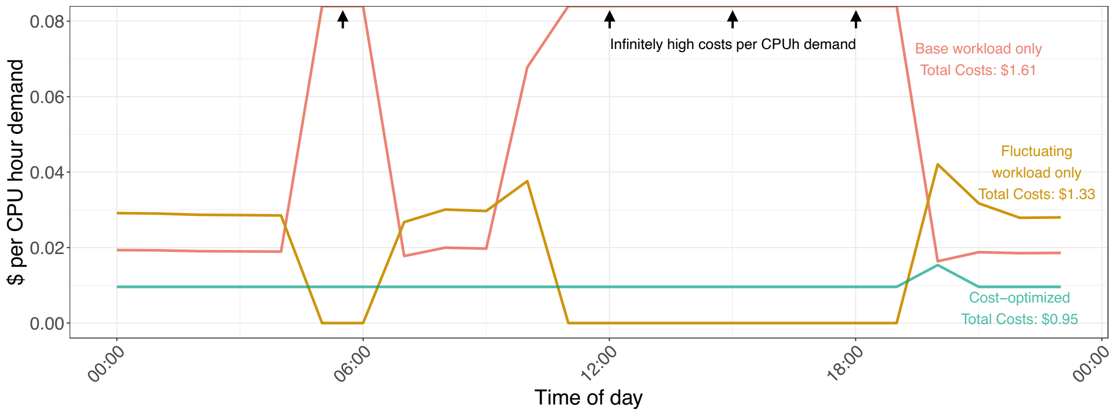
Base workload only. Considering that the workload peak is slightly above 4 CPUh/h and the closest single instance provisions 8 vCPUs, the model (M3) suggests deploying a cluster of seven a1.medium standard RIs with a three-year commitment and an all upfront payment. However, since almost half of the day there are no computing resources required, this option yields daily costs of \$1.61.



(a) VH-3 - Workload demand



(b) VH-3 - Cost-optimized instance configurations and price plans



(c) VH-3 - Costs per hourly CPUh demand for different model approaches

Figure 4.4: Workload demand of VH-3 (22-02-2018), instance configurations and price plans for the cost-optimized approach, and relative cost distribution.

Fluctuating workload only. Given that there are no workloads during certain time periods of the day and total workloads demands are low, it is reasonable to assume that utilizing only flexible spot instances may be close to a cost-optimized solution as computing resources are only purchased when they are actually required. Assuming 0.5 CPUh/h migration costs, daily costs are reduced to \$1.33. Nevertheless, while no costs incur during time periods without any demand, costs are only reduced by 17%. This is the case because a1 instances are deployed for both options. However, discounts for a1 instances are larger for committed price plans than for spot instances. This can be observed in Figure 4.4c as the base workload-only approach mostly has lower per demand costs when there are workloads required. Of course, during time periods with no demand at all, costs are still incurred for the base workload-only approach, resulting in overall higher cost-efficiency by only relying on spot instances.

Cost-optimization. Considering that only a1 instances are utilized in both previous approaches and that the highest workload spike is slightly above 4 CPUh/h, it is obvious to set the base workload to either 4, 2 or 1 CPUh/h, as those demands can all be covered by a single a1 instance, which can then be fully utilized. Applying this, the model calculates respective daily costs of \$1.43, \$1.14, and \$0.95. Thus, in this particular workload scenario, it is most cost-effective to cover almost all workloads with one a1.xlarge standard RI (three-year, all upfront), and the remaining workload spike with one a1.medium spot instance (see Figure 4.4b).

It is noteworthy that the base workload in this approach is only marginally adjusted compared to the base workload-only approach, yet saving costs of up to 41%. This highlights the importance of optimized resource utilization for the base workload and leveraging suitable spot instances for the remaining workloads. Moreover, the model's capability of evaluating and respecting the available cost savings for all potential pricing options on different instance types in determining a cost-optimized solution is emphasized.

4.2 Discussion

The evaluation of the proposed model with real-world workloads from three different Snowflake virtual warehouses (VH-1, VH-2, and VH-3) has provided valuable insights into the effectiveness of the model in handling diverse workload scenarios. The evaluation aimed to approach cost-optimized instance configurations and price plans for executing the workloads of each virtual warehouse, considering different options such as base workload-only, fluctuating workload-only, and a combination of both. The results of the evaluation indicate that relying solely on base or fluctuating workloads is not the most cost-efficient solution in any of the evaluated scenarios. On the one hand, the base workload-only approach, while providing adequate processing power for all workloads, mostly results in a significant waste of computing resources and thus high costs. On the other hand, relying exclusively on spot instances achieves better workload coverage but still at relatively high costs, in particular for larger workload demands. Ultimately, the cost-optimized solution lied in a combination of base and fluctuating workloads for all three virtual warehouses, while the specific ratio with which the model is applied to achieve this optimization varied from use case to use case.

So far, the model has been applied to various workloads using only its required parameters. However, as explained in Section 3.7, the model allows for the adjustment of these parameters. In addition to the base workload, fluctuating workload, and migration costs, there are several optional parameters that can be adjusted to meet specific requirements (see Table 3.1). By altering these parameters, it can be analyzed how different price plans, payment options, and commitment terms affect the cost-efficiency of the proposed model. Furthermore, it can be observed how instance configurations change when adjusting the parallelization fraction as well as when excluding ARM-based instances from the results. Besides the optional parameters, it is also discussed how different ways of obtaining spot prices influence potential instance configurations and how workload growth can be effectively handled with regards to cost-optimization.

4.2.1 Base workload pricing options

When the optional parameters are set to their default values, the model considers all available pricing options to determine the most economic solution for a specified base workload. In this case, the best pricing option will always be either a standard RI or an EC2 Instance SP, both with a three-year commitment and an all upfront payment. EC2 provides the largest discounts for these options as they are the least flexible and require the strongest commitment, both in terms of time and money. However, it may happen that customers have specific requirements and constraints that necessitate variations in these parameters. For instance, some customers may prioritize flexibility and the ability to switch to different instances at any time. Others may have budget constraints that prevent them from making a large upfront payment or committing for three years. In such cases, the parameters of the model can be adjusted to reflect these requirements. In the following, VH-2 is used to consider the implications for the model outcomes in similar scenarios.

In the cost-optimized solution for VH-2, total daily costs of $7.02 incur, utilizing one a1.2xlarge instance in a three-year, all upfront EC2 Instance SP to cover 8 CPUh/h base workload. Now assume that the customer using VH-2 is only able to commit for a one-year period. By adjusting the respective parameter of the model, the best pricing option for the a1.2xlarge remains a standard RI with an all upfront payment, but with a one-year commitment. With the fluctuating workloads remaining the same, this limitation increases the daily costs by almost 15% to $8.06. By introducing an additional limiting parameter specifying that only a payment by installments (i.e., no upfront payment) is possible, the daily costs further increase to $8.27, making it approximately 18% more expensive compared to a three-year commitment and an all upfront payment.

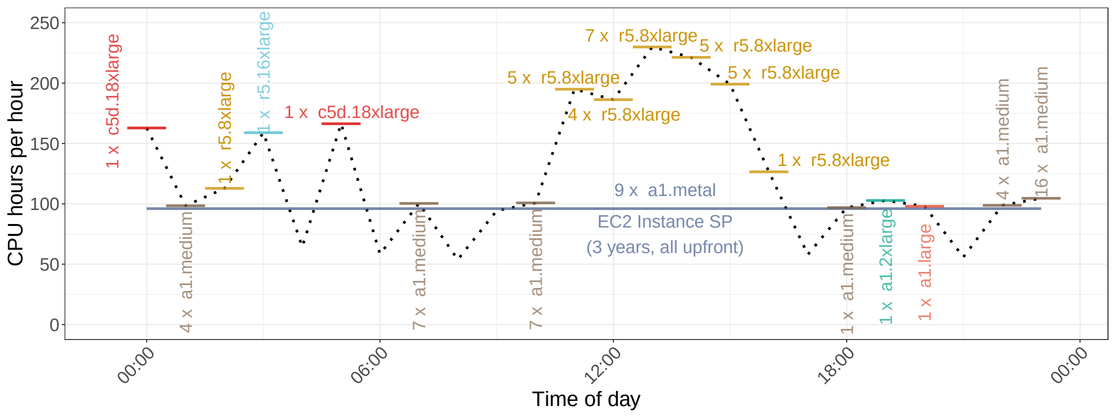
In comparison, assume that commitment term and payment option are no constraints, but the customer of VH-2 seeks the ability to use EC2 instances alongside AWS Fargate and AWS Lambda within the same price plan. Additionally, the flexibility of automatically applying discounts if other applicable resources are used and the ability to switch instance families are important [21]. These constraints mean that Compute SPs must be used for the base workload. Limiting the model with these restrictions results in daily costs of $7.39. This represents 5% higher costs compared to an option where an EC2 Instance SP is used.

Combining all the parameters mentioned above (i.e., Compute SP with a one-year commitment and a no upfront payment), the daily costs increase by over 24% compared to the original cost-optimized solution. Therefore, being able to engage in a long term commitment or willing to pay the full price upfront is crucial from an economical point of view when minimizing cloud spending.

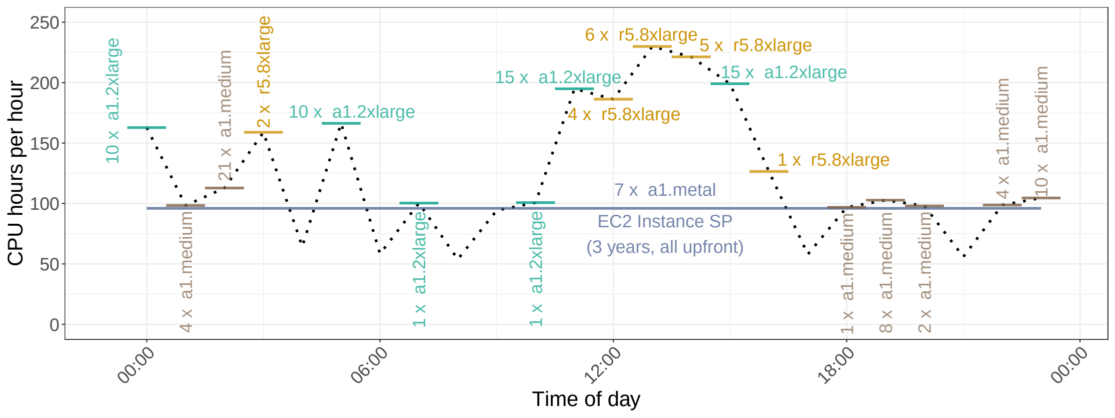
4.2.2 Parallelization fraction

To provide cost-efficient instance configurations more accurately for varying workloads, the parallelization fraction of specific workloads can be adjusted in addition to potential pricing options. By default, a conservative fraction of 0.95 is considered for the model. However, considering the cost-optimized instance configuration of VH-1, the base workload is covered by a cluster of nine a1.metal instances. As a1 instances are specifically optimized for performance and price on scale-out workloads with its custom made AWS Graviton processors [35], parallelization fractions of 0.99 also become realistic. Thereby, the required cluster size for the base workload of VH-1 decreases to only seven a1.metal instances.

Moreover, when encountering workload spikes, larger clusters of spot instances can be employed without getting too expensive, further expanding the usage of a1 instances. Consequently, a parallelization fraction of 0.99 not only reduces the cluster size required for the base workload, but also enables efficient scaling during workload peaks (see Figure 4.5). This ultimately leads to reduced costs, decreasing the daily costs of the workloads of VH-1 to $47.01, saving 16% of the costs compared to the default case.



(a) Parallelization fraction of 0.95



(b) Parallelization fraction of 0.99

Figure 4.5: Cost-optimized instance configurations and price plans for the workloads of VH-1 with different parallelization fractions.

Conversely, if the amount of workloads being parallelized is lower, at 0.90 for instance, the cost-effectiveness of a1 instance clusters diminishes. In such cases, larger c5 instances become a more economic alternative. However, this results in increased daily costs of $59.35. Overall, this highlights the importance of optimizing parallelism in scale-out scenarios in order to reach cost-optimization.

4.2.3 Processor architecture

Even though a1 instances provide the best performance per price ratio for many workloads, especially on small to medium sized workloads, their AWS Graviton processors are ARM-based [35]. Thus, they cannot run software natively that is specifically written for x86 architectures like used in Intel and AMD processors. This can be important for customers who perform lift-and-shift cloud migrations and cannot rewrite their legacy software that may be specific to x86 architectures. To take this into account, the processor architecture can also be set as an optional parameter in the model.

To analyze the impact on the cost-optimized instance configurations, let us consider how the results for the workloads of VH-2 change if ARM-based instances are excluded. The total daily costs increase by 17% from $7.02 to $8.21, as cheaper a1 instances need to be replaced by c5 instances. For the base workload, the best x86 instance is a c5a.2xlarge ($0.116 per hour), whereas the best ARM instance is a a1.2xlarge ($0.077 per hour). This emphasizes the crucial role of building cloud-native systems that run consistently across different architectures. As shown in the example of VH-2, this can reduce costs by providing the flexibility to choose the most economically viable instance configuration without restricting the hardware architecture.

4.2.4 Spot pricing

Unlike on-demand, RI and SP prices, spot prices are not constant. They are rather determined and algorithmically adjusted for each spot instance in each AZ by supply and demand [22]. Thus, spot prices for particular instances can vary greatly over the course of three months. For the evaluation of the model in Section 4.1, the average of the spot prices across three AZs in the last three months were taken as a reference. While this is a balanced way of representing the price of each instance over that period of time, it also leads to some instances being very expensive, sometimes even more expensive than RI or SP options. Consequently, this affects the cost-optimized instance configuration for a particular workload.

A different approach of calculating spot prices is taking the minimum spot price that was available in the last three months, changing the cost-optimized instance configurations for many workloads. This can be perfectly illustrated with the workloads of VH-3.

Taking average spot prices, a fluctuating workload only approach is more expensive compared to the cost-optimized solution where almost all workloads are covered with a RI (see Figure 4.4b). However, using minimum spot prices, relying solely on spot instances becomes the cost-optimized solution as spot prices significantly decrease on many instance types.

Figure 4.6 displays a direct comparison of average and minimum spot prices per current CPUh demand on VH-3 when using only spot instances. In this scenario, during most workload spikes, an a1.xlarge instance is best with average spot prices, and a m4.xlarge instance is best with minimum spot prices. With average spot prices, an a1.xlarge costs \$0.1011 per hour, whereas a m4.xlarge costs \$0.1242 per hour. Conversely, with minimum spot prices, a m4.xlarge (\$0.066 per hour) instance is more cost-efficient than an a1.xlarge instance (\$0.098 per hour). This indicates that most of the time over the last three months and across AZs, the spot prices of m4.xlarge instances were more expensive than a1.xlarge instances. However, during a particular period of time, the spot prices of m4.xlarge were lower than the prices of a1.xlarge.

This demonstrates that a cost-optimized instance configuration is highly dependent on the spot prices available for certain instances during specific time periods. With very low prices, it is more economical to mostly, or even fully, rely on spot instances (as shown for VH-3). However if spot prices for certain instances are rather high due to an elevated demand, it might be the better option to rely mostly on base workloads and cover only the highest workload spikes with spot instances.

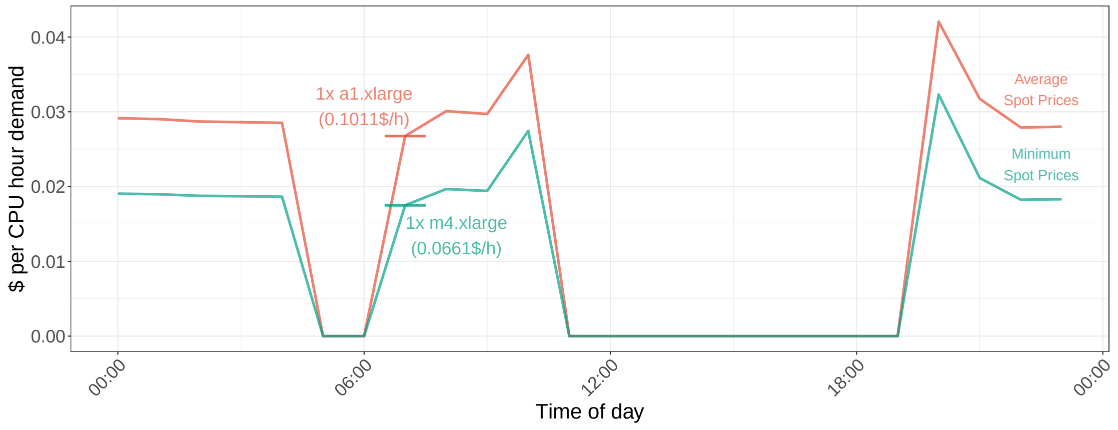


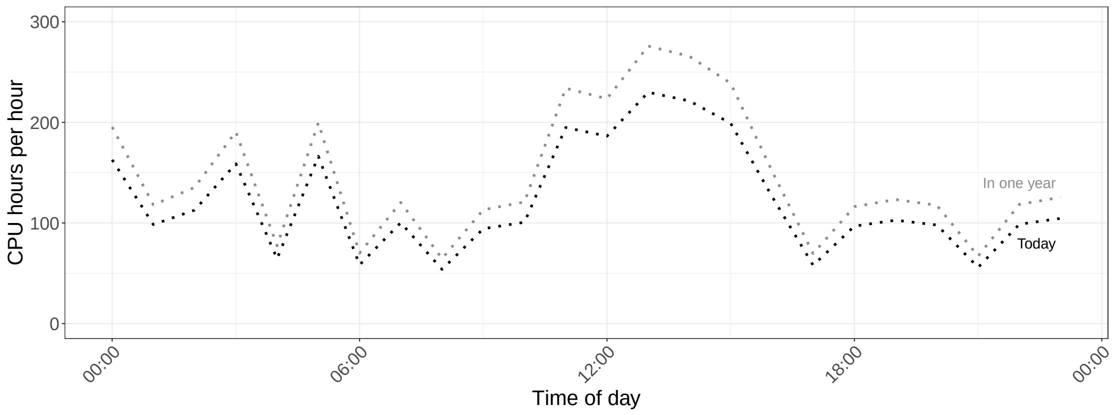
Figure 4.6: Comparison of average and minimum spot prices per hourly CPUh demand for the workloads of VH-3 when using a fluctuating workload-only approach.

4.2.5 Workload growth

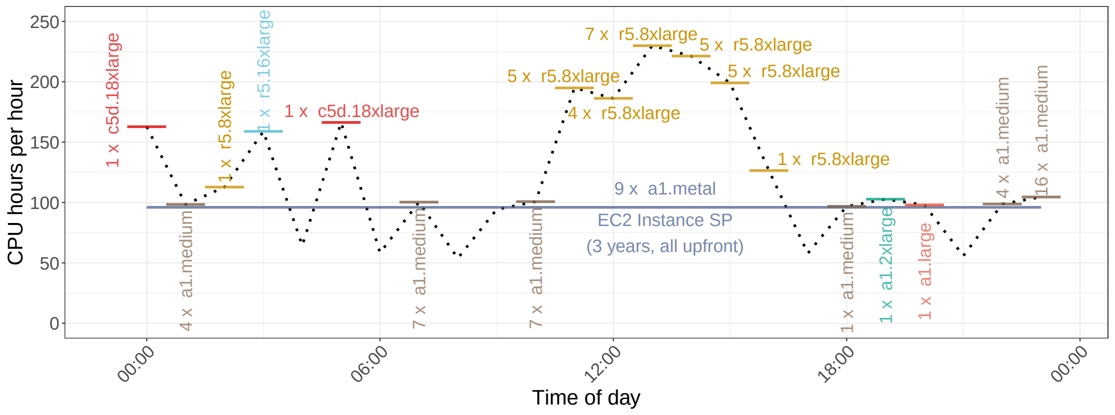
As explained in Section 3.6, cloud customers expecting growth in their workloads are faced with the decision of how to effectively handle it. Two potential options emerge in this scenario. The first involves retaining the instance configurations and price plans recommended by the model for their existing workloads and accommodating any additional workload arising from the growth through the use of flexible spot instances. The second option entails estimating the expected growth rate for a particular time period and recalculating a new cost-optimized solution tailored to these evolving workloads, potentially also changing the base workload. For any of such scenarios, the model performs a comparative cost analysis of these alternatives and offers a recommendation based on the results.

To illustrate this on real-world data traces, it is assumed that the customer employing VH-1 is expecting its workloads to grow by 20% within the next year (see Figure 4.7a). The primary question is whether it is more cost-effective to adhere to the current cost-optimized instance configurations as shown in Figure 4.7b and add up spot instances to accommodate the expected growth or whether it is more economical to also increase the base workload. Given the relatively high growth rate and the already substantial workloads, covering the growth with spot instances proves more expensive compared to expanding the volume of the SP to utilize larger instances for the base workload. Consequently, the instance configurations displayed in 4.7c result in daily costs of $67.41, leading to cost savings of approximately 4% in contrast to daily costs of $70.34 if the base workload were to remain unchanged.

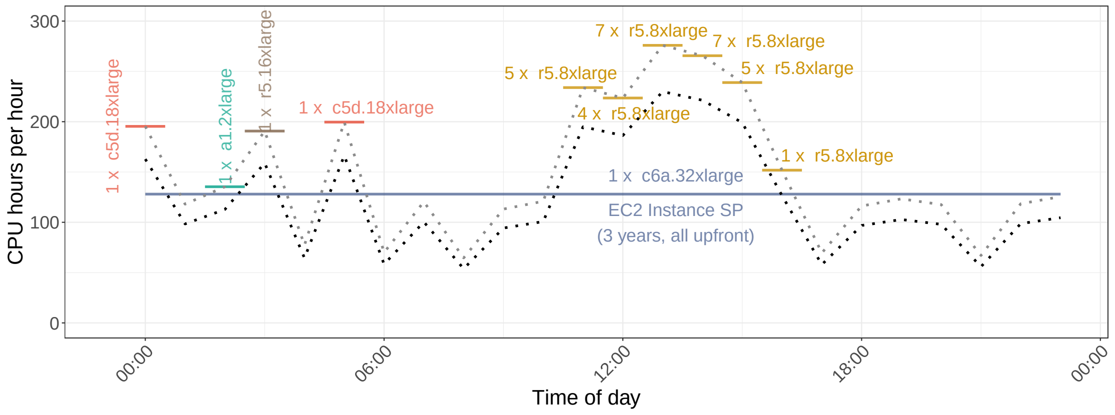
In general, the cost-effectiveness of the two options depends on the specific growth rate and the unique characteristics of the workload demands. Therefore, the more economical approach varies from one use case to another. For instance, with a growth rate of only 1%, it proves more cost-effective for VH-1 to retain the current base workload and incorporate additional spot instances, as the increases in the workload demand remain relatively modest. For smaller warehouses like VH-2, it is also more preferable to employ more spot instances, even when dealing with higher growth rates, as the overall workloads are lower. From an economic standpoint, it is crucial for cloud customers to adopt methods that accurately predict the growth in their workloads. This foresight enables them to make informed decisions and select the most cost-effective solution that aligns with their specific growth and workload requirements. By making data-driven decisions, cloud users can optimize their resource utilization and minimize overall costs.



(a) VH-1 - Current workload and workload in one year assuming a 20% growth rate



(b) VH-1 - Cost-optimized instance configurations and price plans today



(c) VH-1 - Cost-optimized instance configurations and price plans considering growth

Figure 4.7: Model results considering an exemplary growth rate shown on the workloads of VH-1.

4.3 Limitations

The proposed model offers a simple approach to cost-optimization with just three required variables. As demonstrated in Section 4.1, the model effectively delivers cost-optimized instance configurations and pricing options for a variety of real-world workload examples. However, certain workload demands may introduce additional constraints that restrict the flexibility of the utilized instance configurations or pricing options. The model solves this by incorporating the possibility to set optional parameters. By utilizing constraints, the model's outcomes can be tailored to meet specific requirements, ensuring that a cost-optimized solution is achieved while adhering to these limitations. In this way, the model maintains its versatility while allowing for customization to address a broader range of complex scenarios. However, the proposed model still operates based on certain implicit assumptions which are justified in the following.

A priori workload knowledge. One crucial assumption is the availability of inputs in the form of hourly CPUh demands. While cost-optimization often lies in dividing these workload demands between base and fluctuating workloads, it remains essential to have prior knowledge of them. As past work indicates, around 40% of analytical workloads are recurring jobs that are rerun periodically [36], meaning that workload demands for future time periods can be estimated to a certain extent. In addition, the model is designed so that an approximation of the workload demands can also be sufficient. The goal of the model is to provide an efficient overview of the respective cost-optimized solution given a certain workload demand and potential constraints. If actual workloads deviate from an approximation, it can be rebalanced by adjusting flexible spot instances. However, the closer the provided workload demands are to reality, the more accurate the model will be in terms of cost-optimization.

Single node parallelization. As for the particular instance configurations chosen to be cost-optimized, it is assumed that perfect parallelization is achievable on single nodes. For the majority of EC2 instance types that provide fewer than 100 vCPUs, prior research suggests that near-perfect scaling can be achievable [37]. However, when it comes to scale-out scenarios involving clusters of multiple machines, achieving perfect parallelization becomes more challenging. Recognizing this, the model considers the parallelization fraction, acknowledging imperfect scaling that occurs in such scenarios.

Instance hardware. Lastly, instance selection is based solely on the respectively provided vCPUs. Therefore, other potentially important metrics such as network bandwidth or instance memory are neglected, even though they might be important for a cost-optimized instance configuration based on the type of workload that is executed. However, as workloads in this model are characterized only by the amount of CPUh/h, further information on the workload requirements would be needed to appropriately include other metrics. There is extensive research trying to accurately determine and optimize the search for a cost-efficient instance configuration based on the provided hardware specifications given a particular workload [27]. The objective of the proposed model rather is to approach cost-optimization by applying multiple instance configurations to different parts of a workload based on their computing capacity, while maximally leveraging the available pricing options.

5 Future Work

To manage the complexity associated with identifying the most cost-effective instance configurations and pricing options for specific workload demands, certain assumptions - as detailed in Section 3.1 and 4.3 - were made in the context of this thesis. For future work, there is considerable potential to build upon this foundational work and refine these assumptions. Integrating them back into the model would allow for a more comprehensive and accurate representation of real-world scenarios.

Hardware resources. To facilitate the search for a cost-optimized instance configuration and pricing option based on a given workload, workloads are only measured in CPUh/h and instances are correspondingly characterized solely by their provisioned vCPUs. This approach significantly reduces complexity as the model can limit the search of appropriate instances for a workload to one dimension. However, it is essential to acknowledge that, while this method effectively approaches cost-optimization for various workloads, it is not entirely accurate. In practice, other metrics such as the amount of data that needs to be loaded from storage over a network with a specific bandwidth might also be important. A more comprehensive perspective of that was considered by Leis and Kuschewski [8] in their model, in which they accounted for these additional metrics. Integrating such refinements into the current model could enhance its accuracy and applicability in practical situations. As a part of future work, exploring and incorporating such factors would be valuable in further advancing the model's capabilities and ensuring a more precise representation of real-world workload scenarios.

Workload ratio. Another promising avenue for improving the model lies in optimizing its required inputs. While the model effectively determines the cost-optimized instance configurations and pricing options for given workload distributions, it currently requires explicit CPUh/h values for both base and fluctuating workloads as inputs. However, as discussed in Section 4.1, determining the cost-optimized ratio between base and fluctuating workloads can be challenging and may not be immediately apparent. To make the model more practical, a valuable functionality to consider adding is the automated determination of the optimized ratio between base and fluctuating workloads. By algorithmically testing various combinations, the model could identify the most favorable ratio based on an overall workload distribution. This would alleviate the need to explicitly provide the workloads in a specific combination. With that, the model could be combined with previous research trying to effectively predict workload demands and determine instance configurations [27] to optimize the search for cost-efficiency in the cloud.

Migration costs. In addition, the outputs of the current model can be enhanced by improving the calculation of migration costs. Presently, the model assumes migration costs to be constant, disregarding potential variations between actual migrations. In practice, the impact of instance migrations in terms of costs can differ significantly depending on factors like the size of the instances involved. For example, the amount of data stored in memory might differ greatly between instances, thus influencing migration costs. To address this, migration costs could be dynamically calculated based on a percentage of the executed hourly workload amount. Thereby, the accuracy of the model can be enhanced, especially for instance configurations with many spot instances.

Instance clusters. Furthermore, future work building upon the proposed model could focus on improving the coverage of diverse workloads by optimizing the utilization of resources. Currently, the model employs homogeneous instance clusters for scale-out scenarios. However, to achieve greater cost-efficiency and broader coverage of workloads, the integration of heterogeneous clusters might be important. Incorporating heterogeneous clusters containing various instance types or sizes could enable the model to adapt more flexibly to the specific requirements of different workloads. By combining different instances, the model could allocate resources more effectively according to the unique demands of each workload scenario, contributing to further optimize costs.

Data. Lastly, a more comprehensive dataset can be used to enhance the results of the model. While this thesis includes the main pricing options, there are various additional possibilities that could be leveraged to optimize costs. For RIs in particular, a marketplace where unused RIs can be sold and purchased from other customers with heightened discounts is available. Additionally, EC2 offers volume discounts that apply when a specified spending threshold is met, presenting significant potential cost benefits for larger customers.
With a specific focus on spot instances, future advancements of the model could encompass the dynamic retrieval of real-time spot prices. In the context of this thesis, historical spot prices from the last three months were utilized as a reference to determine the most cost-effective instances for fluctuating workloads. However, as spot prices fluctuate with supply and demand, there might be scenarios in which the initially

determined cost-optimized instance configuration becomes less economical under current price conditions. Thus, embedding an automated mechanism to fetch real-time spot prices during the model's execution could yield more accurate and realistic outcomes.
For this thesis and the implementation of the proposed model, EC2 data was utilized. However, adding data from other cloud providers like Azure and GCP would enable a comprehensive cross-cloud analysis in terms of cost-efficiency and could provide valuable insights when approaching cost-optimality.

6 Conclusion

The efficient usage of computing services in the cloud is essential for building cloud-native systems that ultimately aim to approach cost-optimality. For workloads executed on those services, cost-optimization depends heavily on the efficient utilization of different pricing options and the discounts they provide.

This thesis presents a model which optimizes costs for given workload requirements by leveraging the best pricing options for suitable instance configurations on different parts of a workload. While committed pricing options such as RIs and SPs are used to address stable, continuous workloads, flexible spot instances are deployed to handle occasional workload spikes that exceed the steady baseline.

The effectiveness and viability of the model is demonstrated using real-world data traces from Snowflake. By applying the model to workloads of different sizes and patterns, its proficiency to provide cost-optimized instance configurations and price plans in diverse workload scenarios is shown. Depending on the workload characteristics, cost savings of up to 73% can be achieved using the model's cost-optimized solution compared to simpler approaches such as covering all workloads with a single instance configuration used in a committed price plan. Additionally, the model provides the possibility to set multiple optional parameters such as the length of the commitment term for RIs and SPs. They can be used to adjust the cost-optimized solution to the potential requirements and constraints some customers producing the workloads might have. Thereby, the relevance of the model outcomes can be significantly increased.

The proposed model contributes to the objective of approaching cost-optimality on cloud computing services by extending the search for the best instance configurations for given workloads by the effective utilization of available pricing options. While the model operates under certain simplifying assumptions for the scope of this thesis, it can serve as a foundation for future advancements and provides several opportunities for future work to build on.

An implementation of the model using original EC2 instance and pricing data is available open-source at `https://github.com/marcel0096/bachelor-thesis`.

Abbreviations

AWS	Amazon Web Services
Azure	Microsoft Azure
AZ	Availability Zone
CPUh	CPU hour
CPUh/h	CPU hours per hour
EC2	Elastic Compute Cloud
GCP	Google Cloud Platform
OLAP	Online Analytical Processing
RI	Reserved Instance
SaaS	Software as a Service
SP	Savings Plan
vCPU	virtual CPU
VM	Virtual Machine

List of Figures

2.1 Exemplary Snowflake workloads of different virtual warehouses. 8

3.1 Instance configurations and price plans with hourly costs according to model M1-M3 covering an exemplary workload demand (dotted line). . 15
3.2 Illustration for the results of model M4 with regards to workload growth. 17

4.1 Workload demand of VH-1 (22-02-2018) as well as instance configurations and price plans for the cost-optimized approach. 20
4.2 Costs per hourly CPUh demand of three different model approaches for VH-1. 23
4.3 Workload demand of VH-2 (22-02-2018), instance configurations and price plans for the cost-optimized approach, and relative cost distribution. 25
4.4 Workload demand of VH-3 (22-02-2018), instance configurations and price plans for the cost-optimized approach, and relative cost distribution. 27
4.5 Cost-optimized instance configurations and price plans for the workloads of VH-1 with different parallelization fractions. 31
4.6 Comparison of average and minimum spot prices per hourly CPUh demand for the workloads of VH-3 when using a fluctuating workload-only approach. 33
4.7 Model results considering an exemplary growth rate shown on the workloads of VH-1. 35

List of Tables

2.1 Selection of instance families on AWS, Azure, and GCP 4
2.2 Cost savings (compared to on-demand prices) of main pricing options on AWS [17], Azure [15], and GCP [18] 4

3.1 Model Parameters 18

4.1 Hourly cost and savings comparison (rounded) of selected instances . . 24

Bibliography

[1] L. S. Vailshery, *Annual revenue of amazon web services (aws) from 2013 to 2022*, https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/, Accessed: July 18, 2023.

[2] F. Richter, *Microsoft is up in the clouds*, https://www.statista.com/chart/15910/microsofts-annualized-commercial-cloud-revenue/, Accessed: July 18, 2023.

[3] L. S. Vailshery, *Global google cloud revenues from 2017 to 2022*, https://www.statista.com/statistics/478176/google-public-cloud-revenue/, Accessed: July 18, 2023.

[4] S. R. Group, *2020 – the year that cloud service revenues finally dwarfed enterprise spending on data centers*, https://www.srgresearch.com/articles/2020-the-year-that-cloud-service-revenues-finally-dwarfed-enterprise-spending-on-data-centers/, Accessed: July 18, 2023.

[5] T. Robinson, *The cloud backlash has begun: Why big data is pulling compute back on premises*, https://techcrunch.com/2023/03/20/the-cloud-backlash-has-begun-why-big-data-is-pulling-compute-back-on-premises/, Accessed: July 18, 2023.

[6] A. Gupta, D. Agarwal, D. Tan, J. Kulesza, R. Pathak, S. Stefani, and V. Srinivasan, "Amazon redshift and the case for simpler data warehouses," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15, Melbourne, Victoria, Australia: Association for Computing Machinery, 2015, pp. 1917–1923, ISBN: 9781450327589. DOI: 10.1145/2723372.2742795.

[7] B. Dageville, T. Cruanes, M. Zukowski, V. Antonov, A. Avanes, J. Bock, J. Claybaugh, D. Engovatov, M. Hentschel, J. Huang, A. W. Lee, A. Motivala, A. Q. Munir, S. Pelley, P. Povinec, G. Rahn, S. Triantafyllis, and P. Unterbrunner, "The snowflake elastic data warehouse," in *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, F. Özcan, G. Koutrika, and S. Madden, Eds., ACM, 2016, pp. 215–226. DOI: 10.1145/2882903.2903741.

[8] V. Leis and M. Kuschewski, "Towards cost-optimal query processing in the cloud," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1606–1612, 2021. DOI: `10.14778/3461535.3461549`.

[9] M. Villalba, *Aws lambda: Resilience under-the-hood*, `https://aws.amazon.com/blogs/compute/aws-lambda-resilience-under-the-hood/`, Accessed: July 18, 2023.

[10] A. Aithal, *Under the hood: Aws fargate data plane*, `https://aws.amazon.com/blogs/containers/under-the-hood-fargate-data-plane/`, Accessed: July 18, 2023.

[11] AWS, *Amazon ec2*, `https://aws.amazon.com/ec2/`, Accessed: July 17, 2023.

[12] J. Barr, *Choosing the right ec2 instance type for your application*, `https://aws.amazon.com/blogs/aws/choosing-the-right-ec2-instance-type-for-your-application/`, Accessed: July 20, 2023.

[13] AWS, *Best practices for ec2 spot*, `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-best-practices.html`, Accessed: July 20, 2023.

[14] AWS, *Burstable performance instances*, `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-performance-instances.html`, Accessed: July 29, 2023.

[15] Microsoft, *Linux virtual machines pricing*, `https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/`, Accessed: July 20, 2023.

[16] Google, *Machine families resource and comparison guide*, `https://cloud.google.com/compute/docs/machine-resource`, Accessed: July 20, 2023.

[17] AWS, *Amazon ec2 pricing*, `https://aws.amazon.com/ec2/pricing/`, Accessed: July 20, 2023.

[18] Google, *Vm instance pricing*, `https://cloud.google.com/compute/vm-instance-pricing/`, Accessed: July 20, 2023.

[19] AWS, *Instance type details*, `https://aws.amazon.com/ec2/instance-types/`, Accessed: July 17, 2023.

[20] AWS, *Amazon ec2 reserved instances pricing*, `https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/`, Accessed: July 17, 2023.

[21] AWS, *Savings plans*, `https://aws.amazon.com/savingsplans/`, Accessed: July 17, 2023.

[22] AWS, *Amazon ec2 spot instances pricing*, `https://aws.amazon.com/ec2/spot/pricing/`, Accessed: July 13, 2023.

[23] L. Wei, C. Foh, B. He, and J. Cai, "Towards efficient resource allocation for heterogeneous workloads in iaas clouds," *IEEE Transactions on Cloud Computing*, vol. 6, pp. 1–1, Jan. 2015. DOI: `10.1109/TCC.2015.2481400`.

[24] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'17, Boston, MA, USA: USENIX Association, 2017, pp. 469–482, ISBN: 9781931971379.

[25] C. Hsu, V. Nair, T. Menzies, and V. W. Freeh, "Scout: An experienced guide to find the best cloud configuration," *CoRR*, vol. abs/1803.01296, 2018. arXiv: `1803.01296`.

[26] C. Hsu, V. Nair, T. Menzies, and V. W. Freeh, "Micky: A cheaper alternative for selecting cloud instances," *CoRR*, vol. abs/1803.05587, 2018. arXiv: `1803.05587`.

[27] M. Bilal, M. Serafini, M. Canini, and R. Rodrigues, "Do the best cloud configurations grow on trees? an experimental evaluation of black box algorithms for optimizing cloud workloads," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 2563–2575, Jul. 2020, ISSN: 2150-8097. DOI: `10.14778/3407790.3407845`.

[28] D. Kaulakienė, C. Thomsen, T. B. Pedersen, U. Çetintemel, and T. Kraska, "Spotadapt: Spot-aware (re-)deployment of analytical processing tasks on amazon ec2," in *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP*, ser. DOLAP '15, Melbourne, Australia: Association for Computing Machinery, 2015, pp. 59–68, ISBN: 9781450337854. DOI: `10.1145/2811222.2811227`.

[29] H. Bian, T. Sha, and A. Ailamaki, "Using cloud functions as accelerator for elastic data analytics," *Proc. ACM Manag. Data*, vol. 1, no. 2, Jun. 2023. DOI: `10.1145/3589306`.

[30] M. Vuppalapati, J. Miron, R. Agarwal, D. Truong, A. Motivala, and T. Cruanes, "Building an elastic query engine on disaggregated storage," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, Santa Clara, CA: USENIX Association, Feb. 2020, pp. 449–462, ISBN: 978-1-939133-13-7.

[31] A. van Renen and V. Leis, "Cloud analytics benchmark," *Proc. VLDB Endow.*, vol. 16, no. 6, pp. 1413–1425, 2023.

[32] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring), Atlantic City, New Jersey: Association for Computing Machinery, 1967, pp. 483–485, ISBN: 9781450378956. DOI: `10.1145/1465482.1465560`.

[33] D. Comer, *The Cloud Computing Book: The Future of Computing Explained*. CRC Press, 2021, pp. 64–65.

[34] AWS, *Spot instance advisor*, `https://aws.amazon.com/ec2/spot/instance-advisor/`, Accessed: July 03, 2023.

[35] AWS, *Instance type explorer*, `https://aws.amazon.com/ec2/instance-explorer/`, Accessed: July 17, 2023.

[36] S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J. Zhou, "Reoptimizing data parallel computing," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, San Jose, CA: USENIX Association, Apr. 2012, pp. 281–294, ISBN: 978-931971-92-8.

[37] V. Leis, P. A. Boncz, A. Kemper, and T. Neumann, "Morsel-driven parallelism: A numa-aware query evaluation framework for the many-core age," in *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, C. E. Dyreson, F. Li, and M. T. Özsu, Eds., ACM, 2014, pp. 743–754. DOI: `10.1145/2588555.2610507`.