Bachelor's Thesis

THE IMPACT OF WORKLOADS ON PERFORMANCE OF CONFIGURABLE SOFTWARE SYSTEMS

ALEXANDER DINCHER

November 30, 2021

Advisor:
Christian Kaltenecker Chair of Software Engineering

Examiners:
Prof. Dr. Sven Apel Chair of Software Engineering
Prof. Dr. Jens Dittrich Big Data Analytics Group

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

UNIVERSITÄT
DES
SAARLANDES

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,_____ _____

(Datum/Date) (Unterschrift/Signature)

ABSTRACT

Almost every software system comes meanwhile equipped with numerous configuration options that influence the performance. Moreover, the influence on the performance of different configuration options or interactions among them can change over time. For this reason, we investigate the impact of different workloads on configurable systems. In this regard, we look at the number of performance changes and stability of each configuration, as well as the individual configuration options and interactions of them. In doing so, we use performance-influence models to represent the individual influences of the configuration options. In this thesis we analyze the performance behavior of the case studies Tar and Clasp. We found that the workload differences had an impact on both the number of performance changes and stability in both domains. Thus, we could see performance changes of about 78% in the configurations and 19% in the configuration options and interaction of them of the case studies. Furthermore, we found that the influences vary greatly depending on the distribution of performance-intensive configurations and configuration options. Thus, we find a lower number of changes and correspondingly higher stability when the influential terms do not differ across the different workloads. Conversely, we found increased performance changes with lower stability.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Nowadays, it is hard to imagine life without complex software systems. We even use them in our everyday life without being explicitly aware of it. With complex software systems it is not unusual that there are many possibilities to determine the behavior of a system by many different configuration options. Thus, among other things, also a different run time behavior of the individual systems develops. This way, individual configuration options might have an influence on the run time of the highly configurable software system.
Apart from the configurations options there are still a quantity of further factors, which can affect the run time of a software system. Thus, the executed work of the system also plays a role in the run time.However, it is not clear to what extent the workloads affect configurable software systems. Therefore, we want to investigate the impact of different workloads on the performance of configurable systems.
In doing so, we build on the investigations of configurations options in the context of performance prediction by Siegmund et al. [17]. In their work they describe an approach of a model that can predict the performance of these complex systems. It also represents the individual influences of the configuration options.
In our work, we aim at investigating the influence of different workloads on configurable software systems and the configuration options thereof by using a black-box approach (i.e., we do not investigate the source code). We use the additional representation of the performance-influence models to observe not only the run time of the complete configurations, but also the influences of the individual configuration options of different workloads. This results in several point of view on the influence of different workloads of a software system. When comparing different workloads, we examine how much the run time of the system changes with the same configurations and configuration options in the model and, if so, which configuration options are affected by the workload change.
To analyze the impact of different workloads, we consider two configurable software systems in this thesis. They can be divided into different domains. Clasp is used as a solver to solve various mathematical problems. In doing so, it is specified on different problem classes. The second case study is Tar wich is a compression program.
From the previous research of Siegmund et al. [17] and Dorn, Apel, and Siegmund [5], we know that we need to consider some factors when doing working with performance-influence models, otherwise it may lead to wrong results. To do this, on the one hand we use sanity checks, which verify the measurements of the performance values, and on the other hand we use a VIF analysis, which reviews the measured data for muticollinearity.
Using the results of the case studies, we examine the direct performance values and the stability with the use of correlation of the performance measurements. We compare configurations as well as configuration options of the different workloads with each other.

The thesis is structured as follows:
After this introduction, we first address different related work in Chapter 2. In doing so, we first discuss the underlying study of performance-influence models. Afterwards, we describe and compare 6 different publications with our work. The focus is on the selection and handling of different workloads. As a further point of view, the different treatment of performance and its influence is considered.
Afterwards we get into a necessary chapter to be able to follow the further work. Chapter 3 describes the various fundamentals that will be applied in this thesis. We start with performance analysis and highly configurable systems and work through the previous research of Siegmund et al. [17] to describe the terms feature model and performance-influence models in a more detailed way. Next, we look at other techniques and concepts such as Kendall´s Tau Kendall's Tau. Finally, we highlight the topic of workloads, which plays a major role in this work.
Followed by the background knowledge, we present our research of the thesis. The Chapter 4 leads on the one hand into the research questions. Thereby it becomes clear for which reasons the different questions arise and how we answer them. On the other hand, we get an overview of the experimental setup, which includes, among other things, the metrics of the research questions. Furthermore, the two different case studies - Tar and Clasp are presented. Special attention is given to the feature and workload selection and description of the individual case studies.
In Chapter 5 we first describe the results of the case studies. In doing so, we address the research questions defined in the previous section and answer them based on the measured case studies. After the results, we round out the chapter with a discussion of the current research. Here, important points as well as particular findings are highlighted and interpreted once again.
The evaluation of the experimental setup is followed by threats to the validity of the previous results. Thus, in Chapter 6 we explain the internal and external validation of our research.
To conclude the thesis, in Chapter 7 we round up the research with a summary and give a possible outlook into further research of this area.

2 RELATED WORK

In this chapter, we give a first overview of relevant topics and papers related to this thesis. In doing so, the fundamentals but also the differences between this thesis and related works will be discussed. This chapter is intended to serve as an introduction to working with performance-influence models and to clarify the different ways of looking at workloads in other publications. For this reason, we will look at several research articles in the areas of performance model analysis and workload perspectives.

2.1 PERFORMANCE AND ENERGY CONSUMPTION WITH STATIC WORKLOAD

The paper by Siegmund et al. [17] forms the foundation of this work. Thus, the paper is crucial for this research. It describes an approach to the subject of performance impact of highly configurable systems. Based on the many performance or energy consumption values acquired by measuring only a few configurations, a model is learned that is used for performance prediction. However, this model can also be used to indicate which configuration options are relevant in terms of performance or energy consumption. The paper lays the scientific groundwork with a small test study.

Performance-Influence Models for Highly Configurable Systems

Siegmund et al. [17] propose a way to perform performance prediction of highly configurable systems in an efficient way. As they demonstrate in their work, it is more difficult to assess the run time of the system due to the high number of configurations. Siegmund et al. [17] use a so-called performance-influence model to describe all relevant influences of configurable options and their interactions. They consider the system as a black-box and create such a model with the help of machine-learning algorithms on the base of a set of measurements and the respective configurations. In addition to processing binary and numerical configuration options, the method can be used to predict systems with up to 10^{31} configurations by using fairly accurate models [17]. Performance-influence models provide the ability to find a performance regression that is dependent on individual configurations or to determine the best configuration for the system, even for larger systems. Since the research of Siegmund et al. [17] is the first basis for the investigation of performance using performance-influence models, the paper considers only a limited research framework. Thus, individual models of a system are generated, but they are not yet compared with other performance-influence models. For this, the research framework needs to be extended so that for example different software versions or various workloads can be compared with each other.
Our work represents a follow-up to previous research and deals with the impact of different

workloads on the performance of the systems. The performance-influence models generated by Siegmund et al. [17] play a major role in comparing different workloads.

2.2 PERFORMANCE AND ENERGY CONSUMPTION WITH DYNAMIC WORKLOAD

In this section, we describe related work that is not used as a basis for the thesis, but deals with the general field of research. The focus is on performance evaluation with multiple workloads. The papers examine the performance values of the individual systems with different strategies and vary more or less the workloads to be measured. This gives us first knowledge and guidelines how to proceed with different workloads in this work. The first two papers describe further machine-learning approaches that validate their data on different workloads. Furthermore, in the following papers there are research approaches that use different systems like apps, different programs, or tests as a workload basis. We are interested in their experiences with workloads and how they can be improved.

Sampling Effect on Performance Prediction of Configurable Systems: A Case Study

Alves Pereira et al. [2] investigate different sampling strategies to find a representative set of configurations that leads to a good accuracy of performance prediction models. They use different sampling strategies from a previous paper by Kaltenecker et al. [8] to apply it to different non-functional properties. The performance values here are a useful guide to optimize the compression ratio of the sampling strategies. The focus is on the sampling strategies as opposed to the impact on performance. However, the paper considers different workloads. Here, 14 different videos of the video encoder X264 are examined in the case study. In their research, the researchers warn about the impact of different workloads on the sensitivity of a sampling strategy [2]. In this study, we find important indications that different workloads can lead to changes in the measured system.
The examination of different videos provides a wide range of workloads, but it is unclear by which characteristics the workloads were picked. We have not enough information about the differences and similarities of the workloads. Furthermore we do not find out whether the changes of the workloads also apply to the performance. In this work, we focus on specifying and differentiating the workloads. This way, we expect more insights into the performance behavior of different configuration options using different workloads.

Using Black-Box Performance Models to Detect Performance Regressions under Varying Workloads: An Empirical Study

Liao et al. [12] also describe a black-box performance model concept, which is used to find performance regression for different software versions. They rely on dynamically generated workloads, because in the real world workloads are constantly evolving and changing. To create these dynamic workloads, pre-built function-specific workloads are recombined and partially mixed with each other. This results in new and unique workloads. Further, their approach ensures that the learned model detects performance regression not only on the pre-built workloads but also on unknown workloads. In the paper, the focus is clearly on

the discovery of performance regression. In the process, Liao et al. [12] use a sophisticated system for workload diversification.
A disadvantage that arises from dynamic workloads is that afterwards we can no longer clearly identify which workload part is the reason for the performance change. Thus, the previous workload specialization on certain functional areas of the software can not be used for further knowledge or specific performance problems. Furthermore, the randomly assembled workloads can result in many similar ones. As a result, only a small amount of information about the different behavior of different workloads can be inferred.

Transfer learning for performance modeling of configurable systems: an exploratory analysis

Jamshidi et al. [7] uses several environments to get further insights about transfer learning in performance modeling. The different environments describe, for example, workload or version changes. Furthermore, these changes are very well fanned out and classified. Therefore, there are multiple aspects, such as size, quality, or functionality. This background provides a wide scope for the research, but the focus of this paper is on transfer learning.
Consequently, linear regression models like in the paper of Siegmund et al. [17] are not generated and thus the individual effects of the different workloads are not clear in terms of performance. Jamshidi et al. [7] use the paper to clarify how far performance models can be reused even in different environments. Our work uses a comparable structure and classification of the different workloads. However, the focus of our work is on the comparison of different performance models of the workloads.

Analyzing System Performance with Probabilistic Performance Annotations

Rogora et al. [15] examine in their paper performance prediction based on method analysis. Thereby, they do not examine the complete program but analyze separate methods and parts of the software. To predict performance, Rogora et al. [15] measure individual parts with different workloads and subsequently learn models of the whole system. The paper focuses on performance annotations generated by the learned model, which can be used to predict performance.
This approach is more similar to a white-box process instead of a black-box process, which is used in this thesis. Rogora et al. [15] do not consider the system as a coherent object but divide it into individual parts in order to look at the otherwise unknown inner components. As a result, they lose the general representation of the system and have a hard way to transfer their knowledge to other models. However, the workloads are more customized to the system and the individual parts. Thus, there is less lack of knowledge in the workload selection. Our approach moves a step further to make a more general statement about the impact of highly configurable systems with different workloads. We examine different systems based only on their configurations, their input, and their output. In our work, we treat the system as a black box and do not use any information from the internal behavior.

2.3 ADDITIONAL WORKLOAD PERSPECTIVES

This section is intended to refer to the different types of workloads with the following two papers. Thus, we find hidden workloads in the most diverse systems, which are not discovered as such at the first moment. The following sections give a small impression of the variety of workload and workload selection.

App Parameter Energy Profiling: Optimizing App Energy Drain by Finding Tunable App Parameter

Xu, Hu, and Jindal [18] present in their paper an approach to find energy drains in apps. An energy profiler is used to detect high energy consumption. Here, filters are used to determine individual parameters, which are processed using an energy profiling framework to test different values [18].
So in this research we do not find explicit learning of models or other machine-learning approaches to generate similar models. Accordingly, the understanding of workloads is slightly different from our work. In the paper of Xu, Hu, and Jindal [18] different workloads are mapped by tests with different input values for individual parameters. This throws a very different perspective on working with workloads.

Performance and energy consumption of high-performance computing workloads

Mantovani et al. [13] study the performance and energy consumption of a new high-performance computing (HPC) systems in contrast to a previous system. To analyze the HPC system, three different applications with the same benchmark are examined in the paper. The focus is on the performance and energy consumption of the system. Parallelization and the investigation of scalability play an important role.
In relation to this work, the results of the paper are not relevant. However, the investigation of the HPC system reveals a new way of thinking about workloads. Since all applications use the same benchmark, they are responsible for the difference in performance and energy consumption. Thus, the applications themselves can be seen as a kind of workload that triggers different behavior in the HPC systems. Despite the same application domain, the programs are not compared with each other. This is because the applications are too different even though they have the same benchmark. Therefore, in this thesis, we consider different workloads of a system among themselves and not beyond the system.

3 BACKGROUND

In this chapter, we present all background information which is essential for understanding this thesis.

At the beginning, we start with a short introduction of performance value and configurable software systems in Section 3.1 and 3.2. After that we explain the definition and structure of performance-influence models in Section 3.3 followed by the problem of multicollinearity in Section 3.4. Afterwards we take a look into Kendall´s Tau correlation in Section 3.5. At the end of the chapter in Section 3.6 we introduce the concepts of workloads in our thesis.

3.1 PERFORMANCE

There are several ways to evaluate a software system. One method to describe the software is to look at the performance. There are different approaches to measure this value, but in our work we refer to the run time. A typical definition of the performance is:

$$\text{Performance} = \frac{\text{Work done}}{\text{Run time}}$$

However, this is not meaningful in the sense of the software system and this thesis. Here a single task is measured with the system so that the work would be constant 1. Hence the only factor to determine the performance is the time the system needs to compute a result. Instead of describing the performance by

$$\text{Performance} = \frac{1}{\text{Run time}}$$

we use a similar but easier definition. We declare that the performance is equal to the run time of the software system. Thus, in our study we examine the direct run time of the software systems and use the words performance and run time in the same context.

$$\text{Performance} = \text{Run time}$$

3.2 CONFIGURABLE SYSTEMS

In our context, a configurable software offers the opportunity to adjust multiple configuration options (also called feature) that can change the behavior of the software system. These features can specify special functionality of the system, therefore the performance or result of the software can vary. For example, configuration options of the system TAR can decide which compression algorithm is used or how much to compress. This causes a major impact in the run time of TAR. In our thesis, we concentrate on run-time configuration options instead of compile-time configuration options. Run-time configuration option are specified before the execution of the software started aside from that these options are stated in a configuration file or simply as a command-line parameter. In contrast, compile-time options are choosen during the compilation of the software system.

We characterize two different types of configuration options in our configurable software systems. On the one hand, there are binary options which are enabled or disabled and can not change in another value. In terms of our software system, these options can activate a specific part of the software so that the behavior of the system changes. On the other hand, we have numerical options which are represented by numeric values from a valid data set. As a consequence, the software systems is adjustable and can handle different scenarios with, for example, values that are used to calculate the result of the system. Furthermore the valid data set is not limited to a specific amount of values. Due to that, the modern software is highly configurable which could lead to significant performance changes according to various configurations.
A configuration consists of a set of preselected configuration options. According to the previous studies of Siegmund et al. [17], a configuration represents a function from a set of configuration options to a real number. The set of all configurations are called configuration space and describes all possible combinations for each configuration option. The size of the configuration space increases exponentially according to the number of configuration options. We denote the set of all configuration options as $\mathcal{O}$ and the set of all configurations as $\mathcal{C}$. According to that we can define a configuration $c \in \mathcal{C}$ as a function $\mathcal{O} \rightarrow \{0, 1\}$. For example a binary feature $b \in \mathcal{O}$, $c(b)= 1$ if the feature is enabled and $c(b)= 0$ if it is disabled. For a numerical feature $n \in \mathcal{O}$ and $x \in \mathbb{R}$, applies $c(n)=x$ where x is a valid value for feature n and configuration c.
However most of the time we use a specific subset of the configuration space because some configurations are not allowed from system side. A subsets is specified by constraints which are Boolean expressions that describe whether a configuration is valid or not. For this reason every configuration must satisfy all constrains to be a part of the subset. We use constrains to simulate mandatory or alternative options in the configurations. An example of a boolean expression is Feature A $\wedge$ Feature B $\vee \neg$Feature A $\vee \neg$Feature B. The first one implies that Feature A and Feature B must be active to get a valid configuration. The second one depicts a mutually exclusive relation between the two features.

Feature Selection

As already mentioned in the previous passage, the set of configuration options is preselected. This is caused by several important points which are related to the software system. First of all it is not meaningful to use the complete set of all configuration options. According to the exponential growing size of the configuration space the computational effort is highly inefficient. Furthermore a software system often has a high amount of configuration options but only a small part of it effects the performance or does a bigger software impact. As a consequence of that it is useful to select performance-intensive configuration options. For this reason, special attention was paid to the selection of configuration options in the preparation of the case studies. For this purpose, the documentation of the individual configuration options was consulted. Thus, those were selected that have a possible influence on the performance values.

Feature Models

After feature selection, a feature model is created based on all important configuration options. The model contains the properties and interactions of the presorted options from Section 3.2. As described in Section 3.2, these are used to characterize the valid configurations. Mandatory features are configuration options that must always be active or set in order to get a valid configuration. In turn, optional features can be set freely. Interactions include parent-child relations as well as mandatory or forbidden combinations. A parent-child relation is described by 2 or more features. There is a superordinate feature that is called parent. This must be active, so that the lower-level features, called children, can be activated. This gives the opportunity to group several features. It is possible that the parent feature is abstract and not a real configuration option. Another type of interaction describes the alternative group, which is initially a parent-child relation with a further restriction. In a parent-child relationship, multiple child features can be active in a valid configuration. This is forbidden within alternative groups, so that only one child feature per configuration is valid. This describes an either or relationship. Furthermore, interactions can describe the combination of different features that are not directly connected via a relation mentioned above. These ensure that certain combinations may not be selected or must be selected.

A feature model can be represented as a tree structure, these are called feature diagram. There are separate symbols for the different interactions and features, which describe the structure explicitly. Figure 3.1 describes such a feature diagram of the case study TAR. However, only a part of all configuration options is described here to illustrate the different interactions.

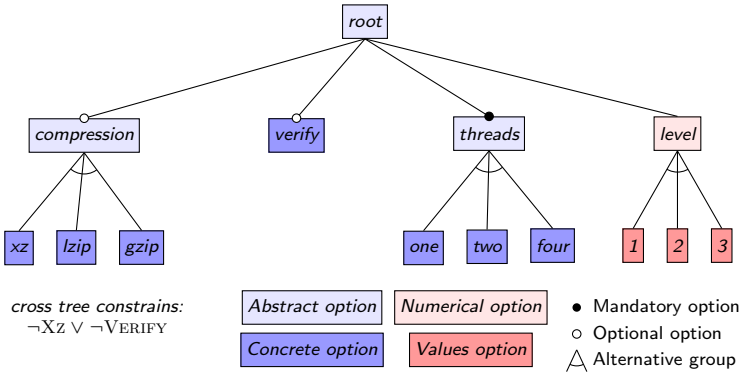

Figure 3.1: A part of system TAR as example for a feature diagram showing features and their interactions

The example of TAR has in total 14 configuration options. These are splitt up in 3 abstract, 7 concrete and 4 numerical configuration options. The abstract feature like COMPRESSION, THREADS and ROOT are not real configuration options of TAR. They link a group of feature to build for example an alternative group like COMPRESSION. The other possibility of binary features are depicted with concrete configuration options. These are called concrete because they describe a real configuration option on the system site and provide functionality. In our example VERIFY, XZ, LZIP, GZIP, ONE, TWO and FOUR are concrete Features. In contrast to binary options the feature diagram in Figure 3.1 has a numerical feature too. LEVEL as

the numerical feature can assume three different values. So the three values 1, 5 and 9 depict the real feature options of the system. Beside the different feature states the diagram shows interactions and constraints as already mention above the example. According to the tree structure we can identify a parent-child relationship in a better way. This is important because a child can only be enabled if the parent feature is enabled as well.
The features VERIFY and COMPRESSION are optional, they are denoted by an edge with a blank circle. In this case we have the possibility to enable and disable this feature. In contrast, THREADS is a mandatory feature as denoted by an edge with a filled circle. In this situation it is necessary that the feature is enabled otherwise the configuration is not valid. The last interaction of the example is the alternative group, which is denoted with dotted lines beginning at the parent feature and ending by all alternative feature options. According to the description of alternative groups it is not possible to activate multiple COMPRESSION features like XZ and LZIP at the same time. This is necessary because the system can not compress data with two different compression algorithms. For this, it is essential to address individual elements of a group without causing a combination of them. Further interactions of systems are not depicted in a feature diagram but as we already mentioned there is the possibility to show cross-tree constrains with boolean expression. An example for the current system is $\neg$XZ $\vee$ $\neg$VERIFY. Thereby both configuration options are mutually exclusive and can not occur together in a valid configuration.

3.3 PERFORMANCE-INFLUENCE MODELS

When considering large software systems, it is unavoidable that the number of configuration options increases. Simply because of the ever-growing user demands which cause additional functionalities and dependencies, the number of configuration options is constantly growing. As a result, the configuration space grows immensely as explained in the previous section. Every single configuration option or combination of them might have an impact on performance, output and functionality. The impact of combinations are the side effects that arise when several configuration options are active together and thereby additional consequences on the system occur, which do not exist with the individual configuration option. For example, it is possible that there are two configuration option A and B, each of them activates a certain type of functionality. However, configuration option A ensures that the functionality of B gets a different value and thus only in combination of A and B a certain result is achieved. Consequently, it is clear that combination of configuration options can have a potential impact. Therefore the user has to keep the whole configuration space in mind to choose the right configuration for his problem. Caused by the high effort, many users take default values or try to make only minor changes. As a result, the software is not used properly and the user does not get the best possible result.

To solve this problem, a so-called performance-influence model is created [17]. Such a model was learned by machine learning and is used to make predictions about the potential run time of a system. The model indicates for all configuration options and relevant combinations how much they influence the run time. Thus, the user can see the impact of individual configuration options and can use the model to select and calculate the required functionality and run time. These models are not restricted to the performance of a system, but they are only used for the run time in this thesis. Siegmund et al. [17] use

a learning algorithm based on multiple linear regression and feature forward selection to create such performance-influence models. They showed in their previous work that this approach is an effective and accurate way to generate these models.

A general representation of a performance-influence model looks as follows:

$$\Pi(c) := \beta_0 + \beta_1 \cdot c(x_1) + ... + \beta_i \cdot c(x_i) + \epsilon \tag{3.1}$$

Here $\Pi(c)$ describes the run time of the system of configuration $c \in \mathcal{C}$. This duration consists of a basic time β_0, an error ϵ and the time influences of the different configuration options $x_1 ... x_i \in \mathcal{O}$. The sets $\mathcal{C}$ and $\mathcal{O}$ are defined in Section 3.2. The base run time β_0 is independent of any configuration option. It describes the execution time when no configuration options are enabled and is taken into account for each configuration. $\beta_j \cdot c(x_j)$ describes the impact of configuration option x_j on the run time. $c(x_j)$ describes if the configuration option x_j is active, see Section 3.2. β_j shows the run time change caused by configuration option x_j. It is possible that the changes due to configuration option j have a positive or negative effect. Furthermore, $\beta_{ji} \cdot c(x_j) \cdot c(x_i)$ can also be used to describe a run time change caused by a combination of configuration option x_j and x_i. Here, the running time β_{ji} is computed in addition to β_j and β_i.

For the previous example Figure 3.1 in Section 3.2, a performance-influence model could be as follows:

$$\begin{aligned}\Pi(c) := {} & 30 + 5 \cdot c(Verify) + 50 \cdot c(Level) \cdot c(Level_5) + 100 \cdot c(Level) \cdot c(Level_9) \\ & + 30 \cdot c(Compression) \cdot c(Xz) - 20 \cdot c(Threads) \cdot c(Four) - 10 \cdot c(Threads) \cdot c(Two)\end{aligned}$$

In this example, the configuration options VERIFY, LEVEL, $Level_5$, $Level_9$, COMPRESSION, XZ, THREADS, TWO and FOUR are relevant and therefore mapped in the influence model. The model contains a basic run time of 30, which TAR requires without an active configuration option. By enabling the VERIFY configuration option, the run time is increased to 35. Furthermore, several run time changes can be detected by configuration option combinations. For example, the user can turn on the configuration option THREADS and TWO and improve the run time by 10. To provide a further characterization of the performance-influence models, Siegmund et al. [17] has developed a machine-learning algorithm that creates such a model based on measured configurations. This algorithm has been integrated into the tool SPL Conqueror [1]. In this work, the tool is used to compute the models so that we can compare them.

3.4 MULTICOLLINEARITY

To compare the different performance-influence models from Section 3.3, we need to ensure that they are consistent and comparable. One issue that arises when using linear regression is multicollinearity.

Multicollinearity describes the linear relationship between two or more variables. There is often confusion with collinearity, which only considers two variables and no linear

1 https://github.com/se-sic/SPLConqueror - last visited on 22.11.2021

combinations. So we can say that collinearity is a special case of multicollinearity [1]. Nevertheless, both terms are used to determine the correlation of different variables. For example, a new variable X may not add any new information to a model, since it can already be represented by the existing variables Y and Z. In this case, the parameters X, Y, and Z have the same information. The variables X, Y, and Z interact with each other. Thereby they bring multicollinearity into the system. Furthermore, it is possible that two or more variables only partially describe the same information. As a result, they cannot be replaced like variable X, however, they correlate with each other. Thus, multicollinearity may occur in different strengths in a system. If no multicollinearity exists, there is no linear relationship between the different variables. Then they are ortogonal to each other [4]. In the state of weak or strong mulicollinearity, several variables are in part correlated to each other. This results in variables that only add a little bit of new informations to the system or model. But as long as they are still an extension to the system, they are still considered. The special case of perfect multicollinearity described above is the possibility to replace variables completely. In this case n variables can be represented by less than n variables. Alin [1] describes the term with another equation but the purpose remains the same, " In more technical terms, multicollinearity occurs if k vectors lie in a subspace of dimension less than k. This is the definition of exact multicollinearity. This is the definition of exact multicollinearity or exact linear dependence." Such cases allow our system to have several different performance-influence models describing the same thing. This means any feature can be defined with different influences. Consequently, the models would lose their comparability, since we would not know which model forms to compare. To solve this problem, we first need to find all features with perfect multicollinearity and clean them up.

3.4.1 *Influence on Measurements & Models*

As mentioned in the previous section, perfect multicollinearity leads to the fact that our models are no longer comparable. So it is important which features are responsible for perfect linearity.

Normal multicollinearity does not play a role here, since it is partially unavoidable by our models. This is due to the fact that some of our features are connected with parent-child relations. Therefore, there is a very high chance that they correlate with each other. However, this does not disturb our performance-influence models at all and is also intended by the relations of the individual features. Thus in the later part of the thesis the general multicollinearity is equated with the perfect one.

Dorn, Apel, and Siegmund [5] present in their paper first insights into perfect multicollinearity in performance-influence models. In this context, they have outlined two possibilities for the occurrence of multicollinearity. On the one hand, multicollinearity develops due to multiple mandatory features in a model as well as in alternative groups. Consider the following example of the system Tar from Figure 3.1 with a subset of features to illustrate multicollinearity.

In the Table 3.1 each line represents a lightweight configuration from the configuration space of Tar. Each column describes whether the feature of the column is active. The last one is an exception. It specifies the performance value of the respective configuration. As we can see the feature Root and Threads are mandatory, because they have to be activated

Table 3.1: Configuration example with preselected features from system TAR to show multicollinearity

root	*Threads*	*One*	*Two*	*Four*	*Verify*	$\Pi(*)$
1	1	1	0	0	0	10
1	1	0	1	0	0	20
1	1	0	0	1	0	30
1	1	0	1	0	1	25
1	1	0	0	1	1	35

in every configuration. In contrast, VERIFY is optional. Furthermore, we know from Figure 3.1 that ONE as well as TWO and FOUR form an alternative group. Here only one feature per configuration can be activated.

Now, all features from Table 3.1 are relevant and are represented in an influence model. Therefore, multicollinearity occurs in the model. On the one hand, this is caused by multiple mandatory features. These ensure that the model cannot assign which influence can be traced back to which feature. Since the base influences of ROOT and the influences of the mandatory feature THREADS cannot be distinguished, the system can divide the combined influence in any ratio between the two features [5]. Thus, the following two models may apply to figure 3.1:

$$\Pi_1(c) := 1 \cdot c(Root) + 8 \cdot c(Threads) + 1 \cdot c(One) + ...$$
$$\Pi_2(c) := 7 \cdot c(Root) + 2 \cdot c(Threads) + 1 \cdot c(One) + ...$$

Consequently, there can only be one mandatory feature because otherwise the influences of the features can no longer be separated and multicollinearity dominates in the system. The other reason for multicollinearity are alternative groups. The reason is that each feature can be expressed by the combination of all other features of the group. For the above example: c(ONE) = 1 - c(TWO) - c(FOUR). This is the definition of perfect multicollinearity. Therefore, the influences of these features are also mixed with each other and lead to multiple models with the same effects:

$$\Pi_3(c) := 0 \cdot c(Root) + 10 \cdot c(One) + 20 \cdot c(Two) + 30 \cdot c(Four) + ...$$
$$\Pi_4(c) := 5 \cdot c(Root) + 5 \cdot c(One) + 15 \cdot c(Two) + 25 \cdot c(Four) + ...$$
$$\Pi_5(c) := 10 \cdot c(Root) + 0 \cdot c(One) + 10 \cdot c(Two) + 20 \cdot c(Four) + ...$$

Since a feature of the alternative group is always activated in each configuration, the influences of the alternative group can also be expressed with the basic influence of the system in each ratio [5]. These factors interfere with the comparability of the models to such an extent that we can only compare them in the cleaned version of the model.

3.4.2 *Elimination of multicollinearity in the system*

The different representation of the performance-influence models can lead to wrong conclusions. Therefore, the models must be cleaned of multicollinearity from Section 3.4.1 so that they remain comparable. The following three subsections deal with the detection of multicollinearity in a system and by which steps the measurements are cleaned up.

Manual observation

First, all features with their interactions and relations can be checked by hand. The focus is on the two factors described above. The search is supplemented by the expertise of the developers. They can identify overlapping features and thereby mark a potential problem area. In the previous studies [17] [5], this task was always done manually without the help of a system. Due to the increasing number of features, interactions can quickly be overlooked. Since multicollinearity can still be secretly introduced into the model in case of human failure, it is recommended to exclude the possibility of multicollinearity by a further test.

Variance Inflation Factor

A much more advanced method is to examine the Variance Inflation Factor. This can be calculated automatically by a system so that no human error contributes to the presence of multicollinearity in the system.
According to Daoud [4], the Variance Inflation Factor (VIF) describes how strongly the variance is influenced by correlation. Multicollinearity causes the standard error of the coefficients to increase. This also increases the variance evaluated by the VIF. The VIF can be calculated as follows:

$$\mathsf{VIF} = \frac{1}{1-R_i^2}$$

The Variance Inflation Factor is computed for each configuration option in our performance-influence model. This is done by a regression of all variables against the i-th feature. This leads to $1 - R_i^2$, which we need to calculate the VIFs of the features. R_i^2 represents the coefficient of determination resulting from the regression of feature i against all other covariates.
The values of the VIF are between 1 and ∞. The following gradations are usually used to describe the correlation of the features [4]:

As described in Table 3.2, a higher VIF results in stronger multicollinearity. However, researchers still disagree on where the exact transition between moderate and high correlation lies. In the table above from the paper by Daoud [4], this threshold is set at a VIF of 5. However, this is not relevant for our work, since we are only looking for perfect multicollinearity in our model. We can discover it by a VIF of ∞ and flag features to clean them up afterwards.
In a later step, the VIF analysis can be used to extend the algorithm to build the performance-influence models. In this process, not all relevant features are included in the system as before. Furthermore, the features do not have to produce a perfect correlation in the existing

Table 3.2: Variance Inflation Factor explanation

VIF Value	*clarification*
1	*not correlated*
$1-5$	*sparsly/moderately correlated*
>5	*highly correlated*
∞	*perfectly correlated*

model in order to be included in the model. This can prevent human errors as well as additional data cleaning work.

Data cleaning

Dorn, Apel, and Siegmund [5] describe in their publication how to deal with feature that generate multicollinearity. This solves the multicollinearity problems for the detected feature from the previous sections. There is no further distinction between an alternative group and a mandatory feature. Both feature types are removed from the model. This has different reasons:
On the one hand, the basic influence and always activated features cannot be kept apart. So it does not make sense to split the influence between different features. This removes all mandatory features so that there is only one bundled ROOT influence. In our performance-influence model, we can illustrate the same behavior without removing the feature. Instead, we set the influence of all mandatory features to 0. This gives ROOT the total influence of all always active features.
On the other hand, we need to remove a feature from each alternative group or set its influence to 0, so that the multicollinearity of the other alternative group features disappears. Here we choose randomly or by domain knowledge a feature which is handled as default option [5]. If this option is removed from the model, the other features can no longer be represented in terms of perfect multicollinearity.
If we look at the previous example from Section 3.1, we find both multicollinearity by mandatory feature and by alternative groups. In this scenario it is necessary to remove the mandatory feature THREADS or set its influence to 0. This way, the two models Π_1 and Π_2 turn into the generalized model Π_6. As a second step we select the feature ONE as a default option of the alternative group and remove it from the model or set its influence to 0. So we reduce the different models (Π_3, Π_4, Π_5) to the model Π_7. This way we can ensure that a model is always created the same and does not have different representations.

$$\Pi_6(c) := 9 \cdot c(Root) + 0 \cdot c(Threads) + 1 \cdot c(One) + ...$$
$$\Pi_7(c) := 10 \cdot c(Root) + 0 \cdot c(One) + 10 \cdot c(Two) + 20 \cdot c(Four) + ...$$

3.5 KENDALL´S TAU CORRELATION

The Kendall's Tau correlation coefficient describes how strongly two ordered data sets differ. This allows us to estimate the stability of two compared datasets in the rest of the work. This process is explained in more detail in the experimental setup in Section 4.2.2.It is important that both data sets contain the same data, otherwise no exact correlation between the data sets can be created. Lapata [11] defines the Kendall's Tau correlation as a metric to calculate the distance between a pair of ranked lists with the same elements. Kendall's Tau is calculated as follows:

$$\tau = 1 - \frac{2 \cdot m(r_1, r_2)}{n \cdot (n-1)/2}$$

Here it is assumed that two ranked lists r_1, r_2 with the same objects x_1 ... x_n are compared with each other. n is the number of objects that are sorted. Most recently, $m(r_1, r_2)$ describes the minimum number of adjacent swaps needed to bring rank r_1 to r_2 [11].
The calculated coefficient has values in the range -1 to 1 to represent the ratio of the two ranks. If the coefficient is 1, the rank of both data sets is identical. Thus, in the case of stability, we would speak of a stable and robust ranking between configurations or configurations options. If the value takes -1, we have an inverse ranking. For example, the data is in the first position for dataset 1 and in the last position for dataset 2. Since the coefficient can take all values between -1 and 1, the approximation characterizes how strongly the ranking matches or is inverse. If the value borders close to 0 or also is 0, we speak of a ranking, where we cannot make any statement about the ranking. This is because too much data has improved and decreased its rank. Thus, due to too unstructured behavior in the ranks, we cannot determine a clear approximation to equal or inverse ranks. In terms of stability, we refer to a non-stable behavior. This is because we can transfer the rank order to another system very poorly.

Table 3.3: Ranking example of different configurations and the corresponding Kendall´s Tau correlation

configuations	*Rank A*	*Rank B*	*Rank C*	*Rank D*
c_1	1	5	1	2
c_2	2	4	2	5
c_3	3	3	3	3
c_4	4	2	4	1
c_5	5	1	5	4

In Table 3.3, we illustrate Kendall's Tau correlation using four different rankings, all involving configurations c_1 through c_5. When we compare the two ranks A and B using Kendall's Tau, we obtain a coefficient of -1. This is because the configurations are ordered inversely. Thus, we would assume a non-stable system because the system varies greatly across the different configurations from A to B and we cannot assume that a fast configuration remains fast across different settings. However, things look different for ranks A and C.

Here we find a perfect match of the ranks. So with a Kendall's Tau correlation coefficient of 1, we get back a consistent ranking. Thus, the system is stable over the different versions A and C and we can transfer information about performance from version A to C. In the last case we analyze the versions A and D with their ranks. The completely random ranking of D is immediately noticeable. By examining Kendall's Tau coefficient, we get no additional knowledge in this case. This is because it is 0. Therefore, it is not possible for us to say whether we are approaching a stable ranking. However, we can assume that in this case we do not get a stable model due to the Increased rank changes in the configurations.

3.6 WORKLOADS

Our work take a look at the impact of different workloads. However, we first need to clarify what workloads are, how they have been dealt with until now, what they look like now, and what the difference was between this and the previous research.
To evaluate a highly configurable software system and draw substantiated conclusions from the results, it is necessary to establish a proper benchmark for the system. According to Samuel Kounev and Klaus-Dieter Lange and Jóakim von Kistowski [16], a benchmark is defined as follows:

Definition 3.1 *A benchmark is a tool coupled with a methodology for the evaluation and comparison of systems or components with respect to specific characteristics, such as performance, reliability, or security.*

Furthermore, Samuel Kounev and Klaus-Dieter Lange and Jóakim von Kistowski [16] explains in his book the most important aspects to characterize a benchmark. One of these aspects are the workloads of the system. They describe one or more work tasks that the system has to execute during the benchmark test. It is possible to divide workloads of a system into several separated workloads or to combine several workloads into one larger one. The test simulates the behavior on the basis of the workloads and evaluates it later. In this way, the workloads determine to a large extent how the system behaves. It is possible to run different scenarios and conditions. So the amount of work, but also the intensive search in expensive code segments can play a more important role. In our case of highly configurable systems, there are a lot of functions that are only used with different configuration options or with the help of special workloads. This makes it difficult to show the respective effects on performance. From this short description it is clear how system related the workload selection must be. With inappropriate workloads it is possible that the system cannot execute the work or does not show the required behavior for the respective benchmark. From the related work from Chapter 2 also the different types and ways of workloads becomes more clear. For example, entire software systems are also considered as workloads to properly test a higher-level tool. However, in our research we will stick to basic consideration of workloads for the particular software system. So for each case study we get seperate workloads, which are not comparable across the different systems. Therefore, it is important to study the behavior of different workloads on the run time in terms of performance models.

3.6.1 *Workloads in our system*

This section deals with the rough classification of workloads in our work in contrast to the previous research of Siegmund et al. [17]. As we know from the previous section 2, only one workload was investigated for each system in the previous research. This provided initial understanding of the approach and ensured a well-grounded basis about the performance-influence models. In doing so, the workload was created to be as diverse as possible to the best of our knowledge. This should ensure that it covers the entire functionality of the system. This has advantages and disadvantages. On the one hand it ensures that the entire system with all its functions is considered. However, due to the broad mass of functions, it can ensure that individual functionalities that are represented by certain parts of the workload are lost. Therefore, it is important to consider different workloads of a system and whether they affect our performance-influence models.
In our research, we increase the number of workloads for the reasons mentioned above. We selectively use single workloads to cover each area separately. This is to ensure that the differences between the various functionalities stand out in terms of performance. By splitting a parent workload into several different workloads, we get separate measurements for each workload. Thus, in terms of the performance investigation, Section 4.2.2 does not prevent us from comparing the data.
The workloads are represented in our models as an alternative group of binary features. So each workload corresponds to a configurations option that can be enabled and disabled. Once the respective configurations option is selected, the software is measured with the corresponding workload. The alternative group ensures that only one workload is measured and no workload combinations are created. The parent feature is abstract and is only used to group the workloads.

3.6.2 *Diversity of Workloads*

The different nature and selection of workloads is an essential and important part of this work. It contributes to the generation of well-founded and meaningful results in this research area. It is mandatory to investigate different workloads, otherwise no significant differences in the performance-influence models will emerge. Before we can make a good workload selection, it is beneficial to examine the system under test more closely. It is important to identify different functional areas that can be covered by later workloads. If we take a closer look at a database application, for example, we will find various application tasks. We can first classify these roughly. One application of a database is to read data from it, as, for instance, in digital phone books. Other databases also allow the manipulation of the data. The data manipulation can be further divided into data creation, data editing and data deletion. Thus, in the simplified example, we find four different application areas that can be represented by different workloads. However, this task is very difficult for unknown systems. This is partially because we are looking at the system as a black box. Thus, we do not look at the system directly but only at the behavior that can be observed from the outside. This ensures that we must conclude with the help of domain knowledge or on the basis the configuration options on different application areas. The behavior of a black box in our research is purposefully chosen. Since we want to make a well-founded statement about

performance changes in the most different system with as little information as possible. Further non-functional properties can also be taken into account as additional workload diversity. For example, the size, or the difficulty of the workload, among other things, may cause a different treatment in the system.

4 METHODOLOGY

In this chapter, we present the methodology that we use in our work. Thereby, the chapter includes the research questions, the experimental setup with the measurement properties, the metrics and the case studies to answer the research questions. Thus, in Section 4.1 we address the research questions and in Section 4.2 we introduce the Experimental Setup. In addition to our measurement cluster, we present the measurement configurations. As a further point, we introduce the examination of the performance. Besides the performance values, the stability of the measurements is taken into account. These serve us as metrics to answer the research questions. Finally, we describe the operationalization of the research questions in Section 4.3.

4.1 RESEARCH QUESTIONS

The research questions serve to determine the effects of different workloads on performance. Therefore we investigate multiple perspectives to answer the questions. In our research we have two different levels, as well as two different methods to look at the performance of different workloads. Thus, we can identify four different ways of addressing the situation. Each possibility is examined in a separate research question.
Furthermore, we can divide the research questions into two groups of two research questions each based on their different levels. We find the different performance investigation levels as a part of the next section. Roughly speaking, they describe the analysis of direct performance measurements in the configuration level and the consideration of performance-influence models in the option level. The research questions **RQ1.1** and **RQ1.2** deal with the configurations level. They focus on the performance measurements of all configurations of the systems. They are compared to the performance values of the same system running a different workload to elaborate the differences between the different workloads and to determine the impact on performance. The further research questions **RQ2.1** and **RQ2.2** deal with the second level of investigation, the option level. This deals with the analysis of the learned performance-influence models from Section 3.3. Here, the focus is on examining the individual features with their influences and whether they are affected by different workloads.
As a further distinction of the research questions, the two different methods for performance considerations can be used. We distinguish between the direct comparison of the performance values and the consideration of the stability of the performance values. In the direct performance analysis, which is done in research questions **RQ1.1** and **RQ2.1**, the performance values of the individual configurations are used in the configuration level. In the option layer, the individual influences of features are used to detect performance changes between different workloads. The observation of stability is done in the research questions **RQ1.2** and **RQ2.2**. In this context, the Section 4.2.2 gives a first introduction to the notion of stability. Here, rankings of configurations and rankings of feature influences are

used to make statements about the stability of the system. The stability stands for a rough indication of the performance, which is not dependent on the direct performance values themselves.
Overall, this leads us to the following list of research questions:

RQ1.1: What is the fraction of configurations affected by performance changes between different workloads?

RQ1.2: How stable is the relative performance of configurations in the presence of performance changes between different workloads?

RQ2.1: How frequent and how strong are changes of performance influences of individual configuration options and interactions between different workloads?

RQ2.2: How stable is the influence of configuration options and interactions in the presence of performance changes between different workloads?

4.2 EXPERIMENTAL SETUP

In the following, we describe the general research setup that we use for all case studies. In addition to the hardware and measurement settings, we also describe the metrics of the performance investigation in terms of performance changes and stability of the configurations and configuration options.

4.2.1 *Hardware and Measurement Setup*

The case studies are measured on workstation computers of a cluster. In TAR and CLASP, the cluster has 14 *Dell Optiplex 9020* nodes with a respective Intel Core i5 processor. In the following, these nodes are described as *i5 nodes*. The *i5 nodes* are equipped with an *Intel Core i5-4590* processor with 4 cores and a base frequency of 3.3 GHz. The four cores enable us to measure the case studies with up to four threads. Besides the i5 CPU the PCs have 16 GB of RAM and a 256 GB SSD to store the data of the case study. Furthermore, each of the *i5 nodes* has a 1 Gb/s *Intel Ethernet Connection 1217-LM* network interface card, all connected to a *Superstack 4 5500G* switch with 48 ports. A minimal installation of Debian 10 is used as the base system. Since the system was updated between the case studies, the case study CLASP was measured on Debian 11.
Furthermore, each configuration is measured five times and checked if the relative standard deviation is greater than 10% of the measurements. Then the measurements of this configuration are repeated to avoid a too large fluctuation of performance values. The relative standard deviation is calculated by dividing the absolute standard deviation by the mean value.

4.2.2 *Performance Investigation*

Performance investigation is about comparing performance data. There are several comparison possibilities. At the beginning of the study, the individual configurations were

compared with each other, so that fast or slow configurations could be determined on the base of their feature. In a next step, configurations of two different systems are going to be compared. However, it is important that both systems contain the same features and configurations. Otherwise, the two systems cannot be compared with each other. One way is to investigate different software versions or different software settings such as workloads. In order to create a further possibility to compare the single features, Siegmund et al. [17] created the performance-influence models from Section 3.2. This provided the possibility to investigate the influence of the various features. This was not possible at the beginning of the research of Siegmund et al. [17], because the influences were dependent on the whole configuration.
An essential part of performance investigation is to detect a performance change between two software versions. The software versions are not limited to release versions but can also represent the system with different workloads. Since this work deals with the effects of workloads, these are compared in the context of the software versions. We restrict ourselves to the extension of the workloads and leave it with the same release version. Thus, we ensure that the research space is first increased by one unknown before considering the influences of different release version or both variables in another study. This has the effect that in this work the role of workloads is evaluated independently of other confounding factors. Due to the different possibilities to compare the performance with each other, different comparison layers (called level) are formed for this purpose. In this work two levels are considered. On the one hand there is the configuration level, which compares configurations directly with each other. On the other hand performance-influence models are compared in the option level.

Configuration level

The configuration level is the first level of the performance investigation. The same configurations of both systems are compared with each other. Thereby configuration groups can be recognized which take more or less time by different workloads. Further we can determine with the grouping which configuration option can be the potential reason of the run time change. However, this is only a weak estimation, since we can not say which configuration options influence the run time and to what extent.
First, we have to decide when a relevant performance change has occurred. In the case of the configuration level a relevant change of the performance arises if the following applies:

$$|\Pi_{w1}(c_i) - \Pi_{w2}(c_i)| > \frac{1}{10} \cdot (\Pi_{w1}(c_i) + \Pi_{w2}(c_i)) \tag{4.1}$$

For each configuration $c_i \in \mathcal{C}$ it is decided whether the difference of the performance values $\Pi_{w1}(c_i)$ and $\Pi_{w2}(c_i)$ is larger than one zentile of both values. Here, the values $\Pi_{w1}(c_i)$ and $\Pi_{w2}(c_i)$ denote the performance of workloads w_1 and w_2 at configuration c_i. The limit $\frac{1}{10} \cdot (\Pi_{w1}(c_i) + \Pi_{w2}(c_i))$ was chosen deliberately. By $(\Pi_{w1}(c_i) + \Pi_{w2}(c_i))$ a common basis of both workloads is created. This is necessary since the run time of the different workloads also differ due to their varying scale and difficulty in general. If the performance of w_1 and w_2 is not considered in the boundary, the difference will be too large compared to the boundary. Thus, in each configuration a relevant performance change is detected and we get no information about the system. Therefore it is important to include both factors $\Pi_{w1}(c_i)$

and $\Pi_{w2}(c_i)$. Due to the maximum standard deviation of the measurements of 10%, we also set the limit here at 10% of the sum of $\Pi_{w1}(c_i)$ and $\Pi_{w2}(c_i)$. The detailed description of the measurements is introduced in previous Section 4.2. Using this fixed boundary, we can now detect relevant performance changes between two different workloads.
As already described in the upper section of the chapter, there are also disadvantages to the level which lead us to the next level. Finding performance changes based only on configurations does not always provide optimal results. On the one hand, we do not know if all performance changes have been detected. The reason for this is that different versions of a system can also address different features. This results in very similar performance values that do not lead to any performance changes. If we would look at the individual features, we could find that two features with their performance changes neutralize each other. On the other hand it is partly not possible to recognize by the exclusion method which feature was involved from the performance change. This is especially the case when several features influence the run time. Both problems can be solved by looking at the influence of the individual features instead of the overall influence of the configurations.

Option level

The option level represents the second and more advanced level of the performance investigation. It examines the performance values from a different perspective and thereby ensures that problems from the configurations level no longer occur. As already briefly mentioned above, performance-influence models are compared with each other in this level. Thus, first two performance-influence models are created from Section 3.3. To compare these models, the influences of each feature are compared. In this way, we can directly determine which features were influenced by the different systems. This allows us to eliminate some problems from the previous layer.On the one hand, we know which feature is the reason for a performance change and do not have to laboriously determine the feature with the help of the exclusion procedure. Furthermore, we can also find performance changes that were previously hidden due to multiple influences and divide them into separate performance changes. In this way, we are able to detect performance changes at the feature level. However, our performance-influence models represent a summary of the configurations. Thus, it is also possible that individual key configurations are poorly represented by the model because they deviate from the broad mass of configurations.
As in the previous level, first we must decide when a performance change is relevant at the feature level. It is important how we consider the influences of the feature. In the case of the option level, a relevant performance change occurs when the following two conditions hold:

$$|\Pi_{rel,w1}(feature_i) - \Pi_{rel,w2}(feature_i)| > 2 \tag{4.2}$$

$$|\Pi_{abs,w1}(feature_i) - \Pi_{abs,w2}(feature_i)| > \frac{1}{100} \cdot |\overline{\Pi}_{w1} - \overline{\Pi}_{w2}| \tag{4.3}$$

In contrast to the configuration level, two conditions must be met. This is because we consider the influences in relative as well as in absolute terms. The two views are necessary for different reasons. On the one hand, the relative influences describe how strong a

feature is in relation to the overall performance. This is important because it allows us to estimate how much the feature contributes to the overall run time. On the other hand the absolute performance influence is used to exclude too small and therefore unimportant features. The Equation 4.2 uses the two relative performance influences $\Pi_{rel,w1}(feature_i)$ and $\Pi_{rel,w2}(feature_i)$ of the workloads w_1, w_2, and the respective feature $feature_i$. If the difference in the percentage performance influences is greater than two percent, the first condition to a relevant performance change is given. The second Equation 4.3 uses the absolute influence $\Pi_{abs,w1}(feature_i)$ and $\Pi_{abs,w2}(feature_i)$ of each feature. Furthermore, the mean values $\overline{\Pi}_{w1}$ and $\overline{\Pi}_{w2}$ of the performance values of all configurations of the workloads are used to generate a minimum performance bound. This is created by 1% of the difference of both mean values. If after Condition 4.2 the difference of both absolute performance values is also larger than the just mentioned bound, a relevant performance change of both workloads is present in this feature.

Stability of Performance Measurements

In addition to the investigation of the direct performance values of a system, it is also possible to gather further knowledge about the measured configurations. Another perspective is the stability of the measured configurations. Here, we analyze the variations of the measured system. Since our research focuses on different workloads and their impact on the performance values, we look at how much the different versions differ from each other in terms of performance. In doing so, we observe stability by comparing the rankings of the different versions. In this way, we can see a rough estimate of the performance in addition to the direct comparison of the performance values. Due to the partly strongly varying workloads, it is difficult to define a correct threshold on the basis of the performance. Therefore, it is important not to define the stability based on the performance values directly, but to find a guideline if the system works the same despite differences in the general context of the performance. Since the ranking is a rough estimate of the performance, we use it to analyze the system. So we find out which configurations or feature gets a worse or better rank in contrast to the other version due to a performance change. If the ranking remains the same although the workloads are different, this suggests a robust and stable system. So all configurations or single configuration options have the same rank in the system even if they have different performance values. This means that the fastest or slowest configurations always retain the same significance in a different software setting.
As already explained in the section of the performance investigation, see Chapter 4.2.2, there are two different levels which are considered in this thesis. This gives us the opportunity to analyze stability in a more general view of the configurations as a set, as well as the importance of each feature in the different performance-influence models.
To determine the equality of the different rankings we use Kendall's Tau correlation coefficient in our research (see Section 3.5).

4.2.3 *Case Studies*

To answer the research questions, we examine two different case studies. We consider the solver Clasp, as well as the compression tool Tar where we measured compression and

extraction performance. To stay within the scope of a bachelor thesis, we primarily consider the compression values of Tar. In the following sections, we give an overview of the two case studies and present their selected configuration options and their workloads in detail.

4.2.3.1 *Tar*

Tar[1] is a pack tool. It is used to store files in an archive so that the data can be easily transported and backed up. The individual files are compressed and bound together. The tool has several functions, the main one being the creation and manipulation of the so-called tar archives. The program works with various utilities to compress the data. Among them are Zip, Xz, or Gzip. With these programs, Tar can compress or extract the archives to keep the files as compact as possible. Due to the broad applicability of different files and configurability of Tar, this case study provides a good basis for supporting different workloads on such a highly configurable system.
In the case study of Tar, version 1.30 was measured with seven workloads from different domains. Tar can be addressed directly via the console, so that we do not have to worry about any further disturbing factors within the measurement. The run time is only measured as soon as the actual application of Tar starts. Thus, both the required data and workloads are prepared on the individual nodes in advance and then merged with the other results after the execution of the application. All configuration options were analyzed and finally 25 configuration options were selected and considered according to the already mentioned criteria of the workload selection (see Section 3.2). In addition to that, all features are command-line parameters, which are given directly when the software is started. Furthermore, we have to distinguish between two separate measurement processes. On the one hand, the creation and compression of an archive was measured, and on the other hand, the extraction and unpacking. Therefore, this case study has two different measurements that can be compared. In the following, these versions will be titled as Tar_compress and Tar_extract. Since Tar uses different compression tools, it is important to know the version of each tool, otherwise not all configurations may be available. The case study uses the compression tools Bzip2 with version 1.0.6, Xz with version 5.2.4, Lzip with version 1.21, Lzma with version 5.2.4, Lzop with version 1.03 (LZO library 2. 10), Gzip with version 1.9, Zip with version 3.0 and Zstd with version v1.3.8. Beside the performance the memory consumption and the strength of the compression was measured to establish some connections. Tar with its 25 different feature and seven workloads are illustrated by a feature model with 26 880 configurations.

Features

In Figure 4.1, we describe the 25 selected features in a feature diagram. The same notation is used as in Section 3.2, which introduces the performance-influence models. The workloads are abstracted, since they will be examined in more detail in the next section and the clarity would be lost due to the size of the model. In the feature diagram there is the numeric feature Level. For simplicity, the possible values are displayed in this view as individual features, even though they do not reflect their own configuration options in the software. In

1 https://www.gnu.org/software/tar/ - last visited on 17.11.2021

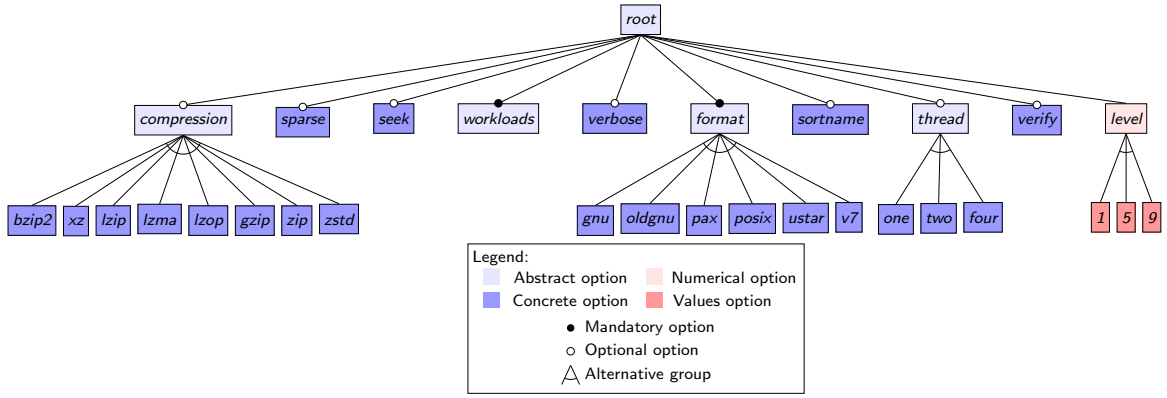

Figure 4.1: The feature diagram of system Tar

the following we describe all selected features and their functions.

compression
: Tar uses different compression parameters to retrieve the different compression algorithms. This feature serves as an alternative group to address the respective ones. However, this feature can also be inactive, in which case only an archive is created or unpacked without using any compression.

bzip2
: Bzip2 ensures that the created archive is filtered by program Bzip2. Thereby the data is transformed with the help of Burrows-Wheeler-Transformation, Move-to-Front-Transformatiion and a Huffman encoding.

xz
: Xz ensures that the created archive is filtered by program Xz. It is based on the Lempel-Ziv-Markow algorithm.

lzip
: Lzip ensures that the created archive is filtered by program Lzip. It is also based on the Lempel-Ziv-Markow algorithm and is therefore on the same level as Xz.

lzma
: Lzma ensures that the created archive is filtered by program Lzma. It also uses the Lempel-Ziv-Markow algorithm like the previous methods.

lzop
: Lzop ensures that the created archive is filtered by program Lzop. It is based on the Lempel-Ziv-Oberhumer algorithm, which is focused on decompression.

gzip
: Gzip ensures that the created archive is filtered by program Gzip. It is based on the deflate algorithm.

zip
: ZIP ensures that the created archive is filtered by program Zip. This procedure deals among other things with lossless data compression.

zstd
: ZSTD ensures that the created archive is filtered by program Zstd. Zstd is a lossless compression algorithm that uses, among other things, fast entropy encoding.

verify
: VERIFY verifies that the archive was written and created correctly. This verifies again after creation that all files are properly present.

sparse
: If SPARSE is enabled, each file is checked for sparseness. If such files are found, they can be given special treatment so that they consume less memory in the archive.

format
: FORMAT specifies the format of the archive. There are 6 different formats and one of them is selected in each configuration. They differ in the allowed file size, file name length, id limitation device numbers.

gnu
: GNU creates an archive in GNU tar 1.13 format. It is basically the same format as OLDGNU, but long numeric fields are handled differently.

oldgnu
: OLDGNU creates an archive in GNU tar 1.12 format or earlier.

pax
: By OLDGNU an archive is created in Pax format.

posix
: POSIX creates an archive in POSIX.1-2001 format.

ustar
: USTAR creates a POSIX.1-1988 compatible archive.

v7
: V7 creates an archive which is compatible with Unix V7 tar. This provides for different constraints, such as the names length of the archive.

seek
: SEEK specifies whether the archive can be seeked.

thread
: THREAD describes the number of cores used to compress and package the data. This feature serves only as an abstract feature to connect the 3 different possibilities. The maximum number of threads is limited to four, because our setup supports only four threads. This feature is optional, because some compression programs do not support this feature. In these cases a default thread count of one is always assumed.

one
ONE describes the use of a thread during the calculations of Tar.

two
TWO describes the use of two threads during the calculations of Tar.

four
FOUR describes the use of four threads during the calculations of Tar.

verbose
VERBOSE can write initial results or information about the work to our console or to a file during run time. This gives the user additional details about the current state of the work that has been done and is still to be done.

sortname
SORTNAME is used to sort the files in the archive. Without this feature the archive is not sorted and has a random order. If this feature is enabled, the files are sorted by name.

level
The LEVEL feature describes how strong the compression should be. Values between 1 and 9 can be specified. The higher the number, the more compressed the files will be. We decided to use 3 different values for LEVEL. So the levels 1, 5 and 9 are distinguished.

Workloads

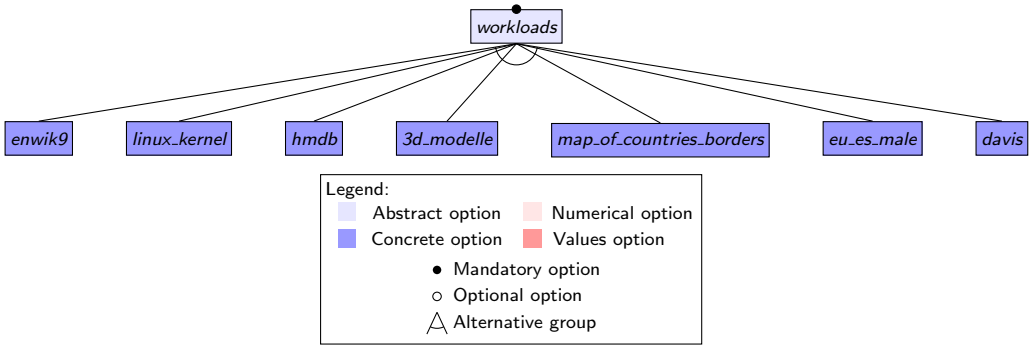

Figure 4.2: The workload feature diagram of system Tar

The seven different workloads examined in the TAR case study are shown in Figure 4.2. The focus of the workload selection was to cover different domains with sophisticated workloads. First, different domains, such as images, videos, different code, music, geo data and 3D models were selected and, as a result, different points of contact were considered. These included compression benchmarks, datasets of the domains, or GitHub. In addition to the type of workload, the size was also taken into account, because datasets from a competition can be very large, so that they go beyond the scope of the case study. In addition to the different application domains, they also consider different compressibility. For this, the selected workloads were pre-measured using TAR and a powerful compression tool Xz on

stronger compression (level 9) to determine the degree of compression. We distinguish between three rough compression levels: easy, medium and hard. Easy files are compressed to about a quarter of the file size, medium files are compressed to about half of the file size, and hard files are compressed to a very small amount. Each workload deals with a different domain. For example, images, videos, different code, music, geo data and 3D models are considered. Thus, we provide a wide division of workloads and completely different files to compress.

Table 4.1: Explanation and comparison of the workload properties of Tar

Workload	*Domain*	*Size*	*Compressibility*	*Original Size*	*Compression Size*
Enwik9	*XTML Text*	*Medium*	*Easy*	953 *MB*	203 *MB*
LinuxKernel	*C Code*	*Small*	*Easy*	185 *MB*	28 *MB*
DAVIS	*Picture*	*Large*	*Hard*	1 050 *MB*	1050 *MB*
HMDB	*Video*	*Medium*	*Hard*	661 *MB*	649 *MB*
3D – Modelle	*3D models*	*Large*	*Medium*	1 350 *MB*	525 *MB*
Map_of_countries'_borders	*Geo (.json)*	*Small*	*Easy*	203 *MB*	36 *MB*
eu_es_male	*Music*	*Large*	*Medium*	1 030 *MB*	513 *MB*

The following Table 4.1 describes the workloads based on their domain, compressibility, and source size. Here, we divide the origin size into small, medium, and large in 500 MB intervals. Furthermore, we briefly introduce each workload in the next bulleted list.

enwik9

ENWIK9[2] is a medium-sized text dataset from a Text Compression Benchmark. It is a part of the english Wikipedia page of 03.03.2006. It is text in XML format and encoded using UTF-8. Because the workload consists mainly of words, it can be easily compressed.

linux_kernel

The LINUX_KERNEL[3] workload contains the source code of the Linux kernel. We are looking at version 2.6.11. The Linux kernel is an open source operating system based on the Unix operating system. The source code is written in C and contains the typical syntax of this code. Based on the size, this workload is interpreted as a small and easy workload.

hmdb

HMDB[4] describes a database of many small videos used for human motion recognition. In its total size, it is normally about 2 GB, but we have reduced it to 660 MB. Thus,

2 https://cs.fit.edu/~mmahoney/compression/textdata.html - last visited on 18.11.2021
3 https://www.kernel.org/ - last visited on 18.11.2021
4 https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/ - last visited on 18.11.2021

only the videos with initial letters b through h were considered in this workload. The videos are stored in AVI format and are divided into 19 categories, each of which contains several videos [10].

3d_modelle
The 3D_MODELLE[5] workload includes several datasets of different 3D models. These are used to investigate 3D mesh compression algorithms. There are 20 different models involved, each generated by several different files. The 3D_MODELS workload, at 1.35 GB, is the largest workload in the Tar case study [6].

map_of_countries_borders
MAP_OF_COUNTRIES_BORDERS[6] is a dataset containing the political country borders of the world. The workload has a resolution of 1m and therefore is 203 MB in total size. The data is stored in JSON format and contains countless coordinates to the borders of the world.

eu_es_male
EU_ES_MALE[7] is a high-quality Basque speech data set. The workload contains excerpted Basque sentences recorded by different people. In our case, we consider the dataset with all male speakers. The files are saved as wave files and do not contain any other folder structure [9].

davis
DAVIS[8] describes an image dataset from 2016, which is used to investigate today's video object segmentation. The images are used to identify objects so that they can be recognized later. The dataset contains over 1 GB of images and is one of the largest workloads we are investigating. The images are all in JPEG format and divided into 480p and 1080p quality levels [14].

4.2.3.2 *Clasp*

CLASP[9] belongs to the genus of solvers. In particular, CLASP is an answer set solver for normal and disjunctive logic programs and uses a conflict-driven nogood learning algorithm. Among other things, this can lead to very good results for sat problems. So it is possible for us to use the software as a SAT solver. The software is written in C++. CLASP has been released as part of the Potassco project under MIT licenses and is available on their site and Github[10].
In the case study of CLASP, version 3.3.6 was measured with 10 different SAT problems as workloads. These were selected from different years of the international SAT Competition[11]. These are described in more detail in the workload section of the case study. The workloads are initially titled very generally as workload 1 to 10 in the model. This is because the

5 https://github.com/abbaselmas/3D-Model-Dataset - last visited on 18.11.2021
6 https://github.com/simonepri/geo-maps/blob/master/info/countries-land.md - last visited on 18.11.2021
7 http://openslr.org/76/ - last visited on 18/11/2021
8 https://davischallenge.org/davis2016/code.html - last visited on 18.11.2021
9 https://potassco.org/clasp/ - last visited on 20.11.2021
10 https://github.com/potassco/clasp - last visited on 20.11.2021
11 https://satcompetition.github.io/ - last visited on 20/11/2021

workloads were still unclear at the beginning of the case study. In this case study, the choice of configuration options had to be made with compromises, since the immensely high configurability of the program means that the measurement duration of the entire configuration space far exceeds the duration of a bachelor's thesis. Thus, a small configuration space with a total of 30 features was chosen, so that a broad spectrum of configurations is covered. Clasp is executed on the console like the previous case studies. In addition to the run time, the memory consumption was also measured, but it was not taken into account due to a too high relative standard deviation in part of the configurations. Clasp has a total of 13 140 configurations by measuring 10 workloads and 30 features.

Features

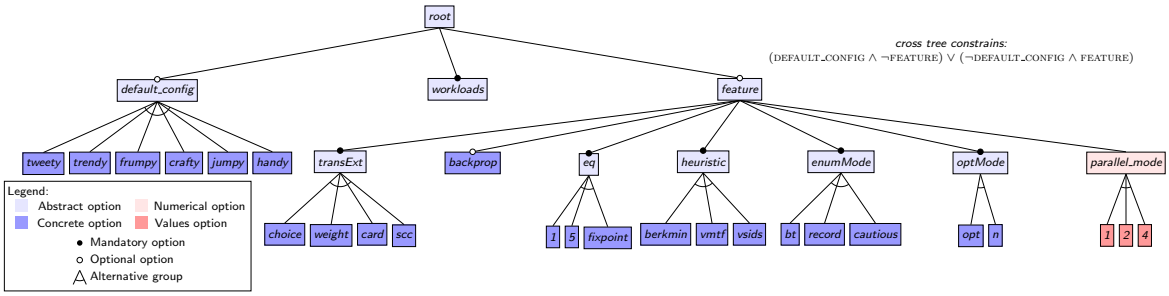

Figure 4.3: The feature diagram of system Clasp

In Figure 4.3 we illustrate the feature diagram from the Clasp case study. As in the previous studies, the workloads are just accumulated and will be discussed separately in the next section. Clasp's feature diagram differs in structure from the other case studies. First, the software provides pre-built configurations that have been optimized based on the developers. These are represented by the Default_config. Our freely configured opions are represented by the Feature group. Either a default configuration or a feature configuration is measured by our defined cross tree constrains (Default_config ∧ ¬Feature) ∨ (¬Default_config ∧ Feature). This way, no pre-built configurations are mixed and we can look at both possibilities. The pre-built configurations use additional features that are not covered by the feature diagram. In the feature group there are many alternative groups with two or three features. Normally, most groups have additional configuration options, but we could only consider a small representative fraction for run time reasons. Furthermore, there is a numeric feature named Parallel_mode, which is represented by an alternative group with its possible values for simplicity. The following enumeration describes in a short way the behavior of each feature. Detailed instructions can be found as a guide under the linked footnote. There, the features are described in more detail.

default_config
: Default_config groups all predefined configurations, but has no functionality of its own.

tweety
: Tweety uses default values to better solve ASP problems.

frumpy
FRUMPY uses old default values which were mainly used in older Clasp versions.

crafty
CRAFTY uses default values to better solve compound problems.

jumpy
JUMPY uses more aggresive behavior to solve the posed problems.

handy
HANDY uses default values to solve very large problems.

feature
FEATURE groups all arbitrary configuration options, but has no functionality of its own.

transext
TRANSEXT describes which extended rules should be converted to normal rules. It groups the different possibilities as an alternative group.

transextchoice
With TRANSEXTCHOICE all choice rules are compiled into normal rules.

transextweight
With TRANSEXTWEIGHT all weight rules are compiled to normal rules.

transextcard
For TRANSEXTCARD all cardinality rules are compiled into normal rules.

transextscc
With TRANSEXTSCC all cardinality and weight rules are compiled to normal rules.

eq
EQ describes the number of iterations that are made when running equivalence reasoning. It groups the different iterations as an alternative group.

eq1
EQ1 takes one iteration at equivalence reasoning.

eq5
EQ5 takes five iteration at equivalence reasoning.

eqfixpoint
EQFIXPOINT iterates on equivalence reasoning until a fixpoint is found.

backprop
BACKPROP enables the use of backpropagation in ASP-preprocessing.

heuristic
HEURISTIC describes which heuristics are used to solve the given problem. It groups the different heuristics as an alternative group.

heuristicberkmin
For HeuristicBerkmin, the solver uses the BerkMin-like decision heuristic.

heuristicvmtf
For HeuristiccVmtf, the solver uses the Siege-like decision heuristic.

heuristicvsids
For HeuristicVsids, the solver uses the Chaff-like decision heuristic.

enummode
EnumMode configures the enumaration algorithm used during solving. It groups the different algorithms as an alternative group.

enummodebt
EnumModeBt enables backtrack-based enumaration.

enummoderecord
EnumModeRecord uses enumaration based on solution recording.

enummodecautious
EnumModeCautious activates the calculation of the cautious consequences of the problem (intersection of all answer sets).

optmode
OptMode describes which optimization statements may be used. It groups the different methods as an alternative group.

optmodeopt
OptModeOpt calculates an optimal model.

optmodeoptn
OptModeOptN first calculates an optimum and then enumerates optimal models.

parallel_mode
Parallel_mode describes the number of threads used to solve the problems. In our case study this feature can take the values 1, 2 and 4.

Workloads

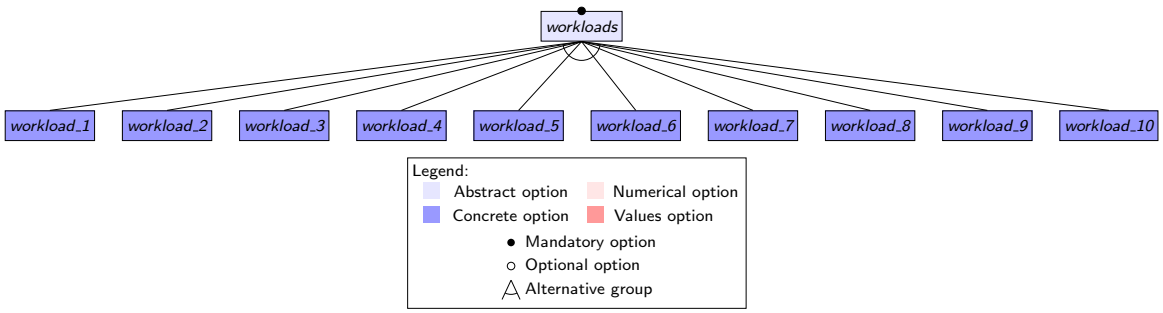

Figure 4.4: The workload feature diagram of system Clasp

The feature diagram in Figure 4.4 illustrates the 10 different workloads in our case study. It is noticeable that, unlike the other case studies, they have very generic names. This is because the selection of matching workloads changed even more frequently at the beginning of the study, as the run time of some configurations in the measurements differed greatly from the previous investigations. In this case study, the workload selection focuses on the criteria SAT/UNSAT problems and the domain. Thus, both SAT and UNSAT problems should be present in the workloads and furthermore, different domains should be covered. Moreover, the workloads should have a minimum run time of 10 seconds, so that unpredictable minimal influences do not take up a significant part of the performance. Therefore, two SAT competitions were examined for the case study. In the first benchmark called SATLIB[12], various problems from different domains were selected and measured using the default configurations. As a result of this, all workloads were under one second, so the competition was excluded for the case study CLASP. As a consequence of that, all selected workloads are from the same set of international SAT competitions. Here, the individual solving problems are divided into different domains. Thus, the normal SAT problems are opposed to the Planning Tracks. The SAT problems are further divided into different problem fields, which are further specified by their domain. Besides the WORKLOAD_6, which comes from the competition of the year 2006, all other workloads are from the year 2020. In addition to the evaluation of the competition, a paper describing the different solvers and benchmarks was published by the creators of the competition. In it we find a more detailed description of the individual problem fields covered by the workloads [3]. In order to select different workloads from this large number of workloads, we first measured all benchmarks from 2020 with a default configuration. Among them were also older benchmarks that were reused from past years. This gave us a first rough estimate of the complexity of the individual workloads. In the next step, we selected about 40 workloads that had a run time between 10 seconds and 30 minutes. These workloads were measured once using all default configurations again to obtain the minimum, maximum and average values of the configurations. Based on these values, the domain and the problem type, 10 different workloads were finally selected to be included in our case study. During the measurements, we found that the developers' pre-built configurations are highly optimized and therefore they differ strongly from the results of the other configurations. As a result, the workload selection were limited to a optimized run time of less then 10 minutes. As a result of that, the following 10 workloads were selected and measured. The workloads are saved as CNF files.

In Table 4.2 we find the final selection of workloads for the case study CLASP. In addition to the division into SAT and UNSAT problems, the domain and the previously calculated values were taken into account. In the description of the individual workloads are the file names of the workloads. Thus the case study can be reproduced despite generic name and if necessary special features of the individual workloads can be made out.

12 https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html - last visited on 28/11/2021

Table 4.2: Explanation and comparison of the workload properties of Clasp

Workload	*Domain*	*SAT/UNSAT*	*Var./Clauses*	*min* $\Pi(s.)$	*max* $\Pi(s.)$	*mean* $\Pi(s.)$
workload_1	*Planning*	*UNSAT*	79506/214309011	151	167	155
workload_2	*Hypertree − decomposing*	*SAT*	200515/22519539	23	191	72
workload_3	*Planning*	*UNSAT*	323352/275157685	205	800	337
workload_4	*Planning*	*SAT*	548457/6169982	43	188	144
workload_5	*Planning*	*UNSAT*	79506/214309011	172	197	180
workload_6	*Traffic*	*UNSAT*	34510/701694	71	137	112
workload_7	*Planning*	*SAT*	277934/152614027	79	359	147
workload_8	*Planning*	*SAT*	72134/112934059	24	41	32
workload_9	*Vlsat*	*SAT*	16676/1598591	10	254	43
workload_10	*Antibandwith*	*UNSAT*	842520/7681710	20	125	46

workload_1

WORKLOAD_1 describes planning workload UNSAT_H_instances_childsnack_p12.hddl_1.cnf which was encoded using Tree-REX.

workload_2

WORKLOAD_2 describes the workload Kakuro-easy-126-ext.xml.hg_7.cnf. This is listed under Main Track Instances and describes a problem based on Hypertree Decomposition. The problems of this class are described on page 91 of the paper by Balyo et al. [3].

workload_3

WORKLOAD_3 describes planning workload UNSAT_P_opt_snake_p06.pddl_30.cnf which was encoded using PASAR.

workload_4

WORKLOAD_4 describes the planning workload SAT_ME_seq-opt_Tidybot_p20.pddl_37.cnf which was encoded using Madagascar E-step.

workload_5

WORKLOAD_5 describes planning workload UNSAT_H_instances_childsnack_p11.hddl_1.cnf which was encoded using Tree-REX.

workload_6

WORKLOAD_6 describes the workload traffic_kkb_unknown.cnf. This is kept under the old part of the main tracks and describes a traffic situation that needs to be solved.

workload_7

WORKLOAD_7 describes planning workload SAT_P_opt_snake_p10.pddl_27.cnf which was encoded using PASAR.

workload_8

WORKLOAD_8 describes planning workload SAT_H_instances_childsnack_p08.hddl_2.cnf which was encoded using Tree-REX.

workload_9
: WORKLOAD_9 describes the workload vlsat2_16676_1598591.dimacs.cnf. This is listed under the main tracks of the competition and belongs to the family of vlsat problems.

workload_10
: WORKLOAD_10 describes the workload abw-N-bcsstk07.mtx-w44.cnf. It is listed under Main Tracks and has formulas describing the feasibility of anti-bandwidth measures. The problems of this class are discussed on page 81 of the paper by Balyo et al. [3].

4.3 OPERATIONALIZATION

This section serves to clarify the research questions and how we intend to answer them through the analysis of the case studies. In the first section of this chapter, we presented the research questions in more detail. Now we describe the approach of the research questions and how we answer them with the support of the case studies.

4.3.1 *RQ1.1: What is the fraction of configurations affected by performance changes between different workloads?*

To answer RQ1.1 we use the direct performance measurement data of the individual configurations of each case study. By using the configurations, we are in the configurations level of our work. To evaluate the measurement data in the context of the research question, we use the presented configurations level metric from Section 4.2.2. Thus, two different workloads are compared with each other and the relevant fraction of performance changes is determined. The process is repeated for each workload combination to obtain a well-founded statement about the impact of the different workloads. To provide a clear answer to the research question, we consider the average proportion of relevant performance changes at this level.
To begin the research question, we want to take a quick look at the performance of all configurations between workloads. By doing so, we will get a general view of the performance distribution of the individual configurations and whether it changes with different workloads. This gives us a first perspective on the data and the impact of the different workloads.

4.3.2 *RQ1.2: How stable is the relative performance of configurations in the presence of performance changes between different workloads?*

In RQ1.2, we continue to use the measurement data from the individual configurations and thus remain at the configurations level. However, in this research question we want to deal with the stability of these values. For this purpose we create a ranking depending on the performance values (see Section 4.2.2). Thus, the fastest configurations of the system are at the top of the ranking and become slower towards the bottom of the list. This ensures that we can compare two ranked lists of configurations without looking at the direct performance

value. Using a rank correlation, we obtain results about the stability of the two workloads, as this is related to the correlation of the two sorted list.
To create a rank correlation between the two workloads we use the Kendall's Tau correlation (see Section 3.5).

4.3.3 *RQ2.1: How frequent and how strong are changes of performance influences of individual configuration options and interactions between different workloads?*

In RQ2.1 we proceed similarly to RQ1.1. However, we now use the data of the learned performance-influence models and look at the performance of the relevant configuration options and combinations of them that are in the models (see Section 3.3). Thus, we are in the so-called option level of the work. To compare the influences of the terms of different workloads, we create a separate performance-influence model for each workload, which contain the same terms. Thereby, the occurrence of multicollinearity has to be investigated and, if necessary, terms have to be removed, as described in Section 3.4. The results of the VIF analysis after data cleaning can be considered in Appendix A.5.
The influences of the configuration options and combinations of them from different workloads are compared in this research question to calculate the relevant portion of performance change. This is defined by the performance investigation of the option level. The metric for this is described in more detail in Section 4.2.2. To provide a clear answer to the research question, we consider the average proportion of relevant performance changes at this level. Before doing so, we also look at the absolute performance influences of all terms between the different workloads to determine influencing terms and whether they remain the same across the different workloads.

4.3.4 *RQ2.2: How stable is the influence of configuration options and interactions in the presence of performance changes between different workloads?*

In RQ2.2, we remain at the option level and consider the influences of the individual configuration options and combinations of them obtained from the performance-influence models. Thus, we continue to use the models from RQ2.1, since they are cleared of multicollinearity. We proceed simultaneously to RQ1.2. We use the influences of the individual terms to create the same ranking as in the configuration level. Here, configuration options and combinations with a high performance influence are ranked higher than terms with a lower performance influence. This allows us to create a comparable ranking despite different levels of influence to determine if configurations options and combinations of them remain stable across different software workloads. After ranking, we use Kendall's Tau to assess the rank-based correlation of these workloads (see Section 3.5).

5 EVALUATION

This chapter describes the evaluation of the case studies and answers the research questions of the thesis. The chapter is divided into two sections. The first section presents the results of the case studies. In doing so, each subsection addresses one research question based on the zwo case studies and concludes with a summary. In the second part of this chapter, we discuss particular findings that came to our attention as we considered the case studies.

5.1 RESULTS

This section focuses on answering the research questions. This involves investigating all of the case studies for each research question and drawing a final conclusion from all of them. In addition to the figures presented in this section, there are additional ones that provide a supporting view of the case study measurement data. They are described in the Appendix as part of the case study specific section (see Appendix A).

5.1.1 *Performance Changes on Configuration Level*

In our first research question, we examine the impact of a workload change based on the performance changes in the configuration level.

RQ1.1: What is the fraction of configurations affected by performance changes between different workloads?

In each case study, we consider not only the absolute performance of all configurations across the workloads, but also the proportion of configurations that differs between the two workloads which are compared, according to the definition of the performance change in this level.

Tar

The following figures describe the compression of Tar and do not address the analysis of Tar_extract. However, the results can be found in Appendix A.3.
In Figure 5.1, we examines the performance of all configurations under different workloads. So we list all configurations on the x-axis and all workloads on the y-axis. The intensity of the color scale describes the performance values of the individual configurations in the different workloads. The same configuration order is used in all workloads. Thus, we can see that the performance changes depending on the workloads in a broad variety of

configurations. The workloads Linux_kernel and Map_of_countries_borders have a small performance margin, whereas the workloads Enwik9, 3D_models, Eu_es_male and Davis have a wider performance range. Furthermore, the workload Davis has a higher number of longer running configurations than all other workloads. Moreover, we can see the similar configuration clusters between different workloads. For example, the workloads 3D_models and Eu_es_male, as well as Davis and Hmdb, and the workloads Enwik9, Linux_kernel, and Map_of_countries_borders behave similarly to each other. For the three groups we find the same performance characteristics, in terms of shorter or longer run time, for the same configurations.

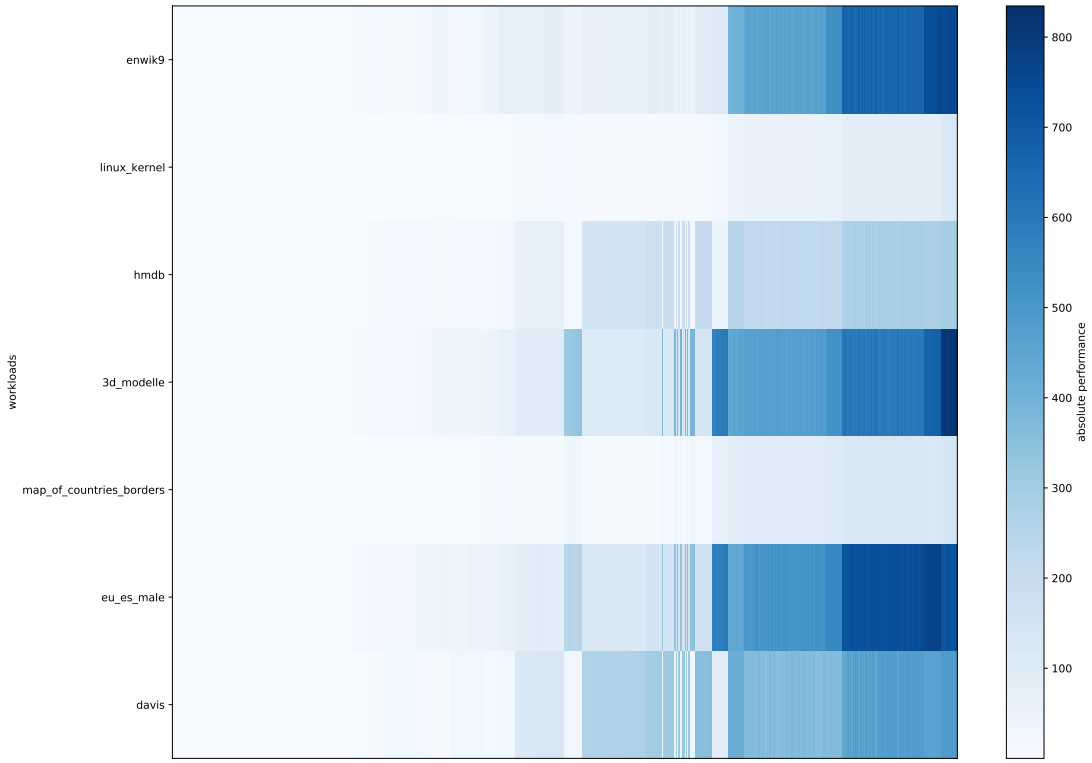

Figure 5.1: Tar_compress performance configuration comparison according to all workloads. All configurations are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective configuration and workload.

If we take a closer look at the results from Figure 5.1, we identify the reasons for the elaborated characteristics. The performance spectrum from the previous description is due to the different workload sizes. Small workloads, such as Linux_kernel and Map_of_countries_borders have a smaller range of performance values and larger workloads, such as 3D_models, Davis and Eu_es_male cause a larger result space or longer run times. This is because the work behaves differently under different configurations. Thus, it is natural that as configurations become more difficult, the run time of smaller work does not increase as much as it does for larger workloads. Next, we analyze the same configuration groupings between workloads. We notice that workloads with the same compressibility

level share the same grouping. The workloads are subdivided as follows. Easy compressible workloads are Enwik9, Linux_kernel, and Map_of_countries_borders. Medium compressible workloads are described by 3D_models and Eu_es_male and finally the hard compressible workloads are Davis and Hmdb. The underlying metric of compression subdivision was described in the workload analysis of the case study (see Section 4.2.3). Thus, selecting workloads based on important system properties, such as compressibility in the case of Tar, provides a large difference in the behavior of the system. So certain configuration parts are easier to compute for hard compressible workloads than for medium or light workloads. This becomes clear when considering the workloads Davis, Eu_es_male, and Map_of_countries_borders. Furthermore, we can see, for example, on the hard to compress workloads Davis and Hmdb, that the performance values of the same configurations differ only based on the workload size. All values increase at about the same rate.

In Figure 5.2, we examine the relevant performance changes that occur between two different workloads. The results of the case studies are represented by an N x N matrix comparing all N workloads. This includes both the percentage of configurations that changed and an indicator color representing that percentage of configurations. The color is used to visually compare the different workload results. The workloads that are compared with each other are located on the x-axis as well as on the y-axis. The percentage of performance changes in this case study is above 50% of the configurations in all except three workload comparisons. Thus, only in the comparison of the workloads Enwik9 and Eu_es_male, Enwik9 and 3D_models, or Eu_es_male and 3D_models do we find a relevant performance change below 50%. Furthermore, the two comparisons Map_of_countries_borders with Eu_es_male and 3D_models stand out. Here, the entire configuration space is affected by relevant performance changes. For the other workload constellations, the performance change rate is between 73.7% and 98.2%. The average number of performance changes between all workloads for the case study Tar is 83%.

By taking a closer look at the knowledge gained from Figure 5.2 and Figure 5.1, we describe them in the following. Due to the high diversity of workloads, we obtain high proportions of performance change in the compared workloads. First of all, this indicates that the consideration of different workloads plays a major role in the performance measurements of configurable systems. In addition to the workload size, compressibility plays a main role in the case study Tar . Among other things, we can see that the workloads that are difficult to compress do not necessarily have the highest run time. One reason for this is that hard-to-compress workloads contain a lot of data that cannot be compressed at all or only to a very small extent. Whereas the medium workloads can be compressed very well, but require a much higher performance. This consideration provides for differences in the workloads. Furthermore, we have to admit that not all comparisons make sense because the workloads are too broad. For example, we can explain the very high performance changes between the workloads Map_of_countries_borders with Eu_es_male and 3D_models and other comparisons. In these cases, very different workloads are compared in terms of size and compressibility. Due to the very large performance differences that consistently occur in all configurations, this also results in a large number of performance changes. Thus, it does not make sense to compare all workloads with each other. The

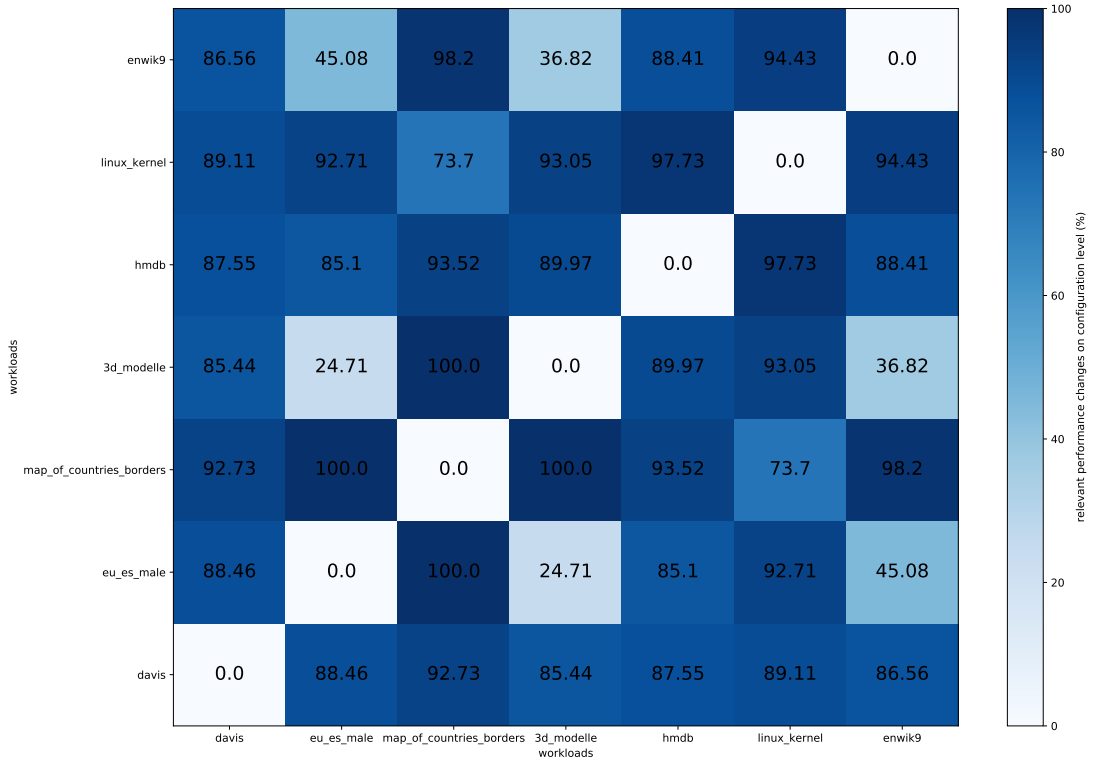

Figure 5.2: Tar_compress relevant performance changes on configuration level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

lowest relevant performance change is obtained by comparing EU_ES_MALE and 3D_MODELS. These are two large workloads that have a very wide performance spectrum. Furthermore, as shown in Figure 5.1, they behave very similarly in the same configurations. This results in a smaller performance difference, which does not exceed the very high limit of a relevant performance difference in this case. However, it does make us aware that workloads with the same characteristics are similar and thus cause a smaller performance change.

Clasp

This section considers the case study CLASP to answer the first research question. In doing so, the workload WORKLOAD_2 has been removed in the following figures for better readability of the other workloads. This is because the workload shows a increased run time in some of its configurations compared to the other workloads. This makes the distribution of the smaller performance values very unclear. Nevertheless, the figures can be found with WORKLOAD_2 in the appendix under the Section A.4.
Using Figure 5.3, we can see the performance of different workloads with the same configurations. For example, we see the same consistently low run time for workloads WORKLOAD_1, WORKLOAD_5, and WORKLOAD_8, among others, in contrast to the other workloads. Among

them are also the workloads WORKLOAD_4 and WORKLOAD_9, which, along with the workload WORKLOAD_2, show an increased run time in a certain part of the configurations. It is especially noticeable that they have their higher performance values in different configuration sections. Furthermore, a similar performance behavior can be observed in the workloads WORKLOAD_3, WORKLOAD_6, and WORKLOAD_7. There, the run time remains constant for the most part across all configurations. WORKLOAD_7 has a slightly increased performance in one third of the configurations compared to the workloads just described. Furthermore, we can see an increase in performance for the workloads WORKLOAD_2 and WORKLOAD_10 in the last third of the configurations. However, the increases are not related to each other, since WORKLOAD_2 increases its performance many times over compared to WORKLOAD_10.

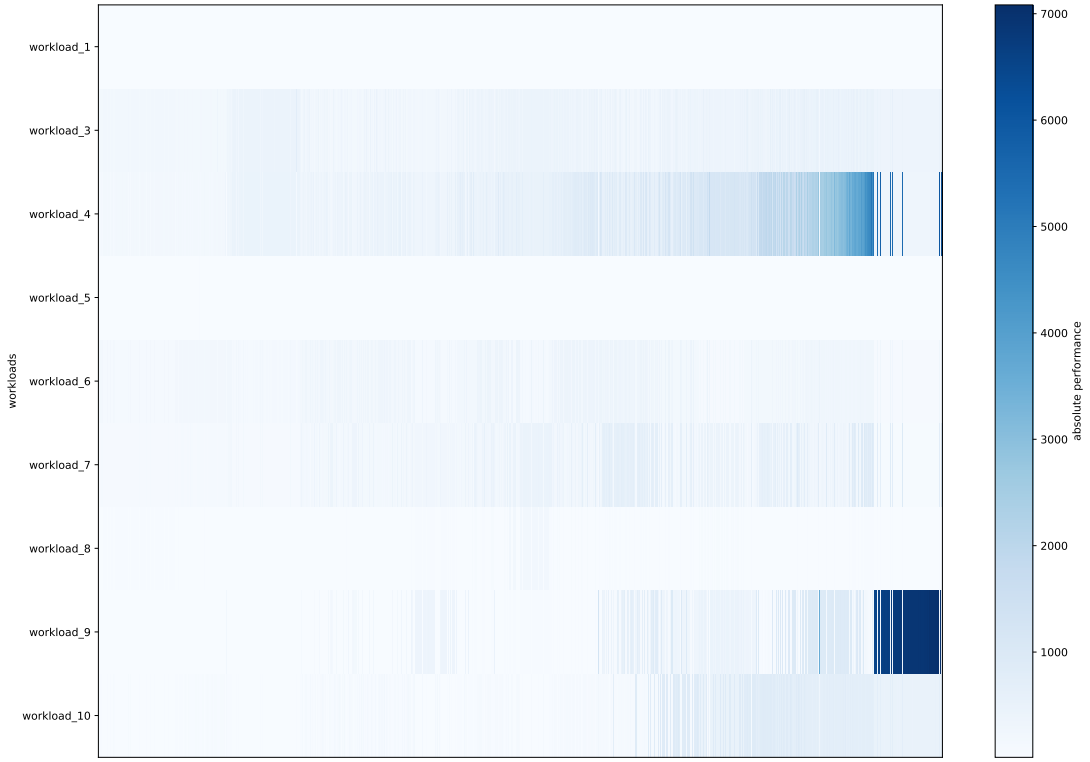

Figure 5.3: Clasp performance configuration comparison according to all workloads. All configurations are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective configuration and workload.

On further analysis of the Figure 5.3 we come to further conclusions with the help of the workload selection. The workloads WORKLOAD_1, WORKLOAD_5, and WORKLOAD_8 describe the same behavior, since they come from the same domain and describe the same problem class. Only the size and the solvability result differ in these cases. Furthermore, we can see that in general the planning workloads have a different run time distribution compared to the other workloads. Likewise, the domains of the other workloads can be identified by their particular run time distribution in the configurations. Thus, for CLASP, we can say that the different domains are characteristic of the run time behavior for the same configurations. The workloads WORKLOAD_2, WORKLOAD_4 and WORKLOAD_9 describe the workload with

the highest performance values. They occur in other configuration ranges. Furthermore, WORKLOAD_10 distinguishes itself from all other workloads by an increase in other configuration sections. This makes the different behavior of the individual domains even clearer.

In the next Figure 5.4 we now observe the actual relevant performance changes that occur between the workloads. It is immediately obvious that we did not find any relevant performance changes for workload WORKLOAD_1 and WORKLOAD_5 in addition to the comparisons of the same workloads. However, we can generally say that a very high percentage of performance changes was also found in this case study. Thus, twelve comparisons alone have a performance change of 100%. Otherwise the values are between 64.08% and 99.85%. The lower performance changes are found in the workload comparisons WORKLOAD_1 and WORKLOAD_8, WORKLOAD_5 and WORKLOAD_8, WORKLOAD_3 and WORKLOAD_4, and WORKLOAD_3 and WORKLOAD_6. The average number of performance changes between all workloads for the case study CLASP is 73%.

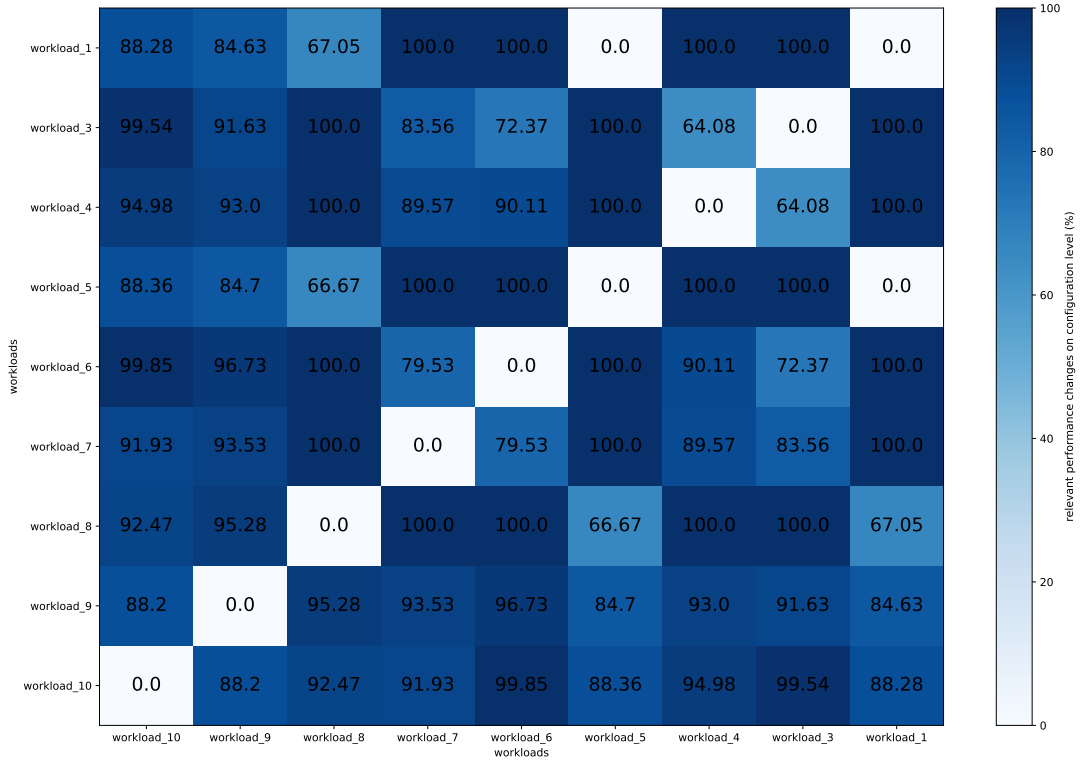

Figure 5.4: Clasp relevant performance changes on configuration level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

By closer examination of the Figure 5.4 we find comparable behaviors as in the previous Figure 5.3. Thus, the workloads WORKLOAD_1, WORKLOAD_5, and WORKLOAD_8 have a lower percentage of performance changes due to their same task domain. In the case of WORKLOAD_1 and WORKLOAD_5, the different workloads are even so similar that no relevant

performance changes can be found. The different solvability of the workload WORKLOAD_8 results in clearly visible performance changes in the comparison. The two workloads WORKLOAD_3, WORKLOAD_4 describe two different problem classes of the planning workloads, but behave relatively the same in the context of the performance, so that also here a smaller portion of performance changes arises. When examining the high performance changes, the two groups WORKLOAD_1, WORKLOAD_5, WORKLOAD_8 and WORKLOAD_3, WORKLOAD_4, WORKLOAD_6, WORKLOAD_7 stand out directly. As soon as a pair from the groups is compared, we find a relevant performance change of 100%. At the same time, all except WORKLOAD_6 are planning workloads. They seem to differ so much based on their problem classes that they have a strongly changing performance compared to each other. The workloads WORKLOAD_9, WORKLOAD_10, and WORKLOAD_2 have a very high percentage of performance changes compared to all workloads. The reason for this are the different domains already indicated before. They provide for a difference in performance in the individual configurations.

Overall

Summarizing RQ1.1, we can say that a relevant performance change of about 78% on average between the different workloads can occur if the workload is highly diverse. Here, we describe high diversity by changing several factors of the workloads. For example, size, complexity, domain, or format, among others, can play an important role when considering different workloads on performance. In addition to high diversity, with a very low diversity, we found through the CLASP case study that the relative performance change can be about 0% when examining different workloads. This makes the analysis of such identical workloads obsolete, as they show no impact in the context of performance.
Through the treatment of the case studies TAR and CLASP it became clear that the performance of a software depends on the characteristics of one factor in particular. This differs from the application area of the system. In the case of TAR it is the compressibility of the workload and in the case of CLASP special attention must be paid to the domain with its problem class. On the one hand they provide for differences in the performance with different configuration areas. Furthermore, they have a greater influence on the relevant performance changes of the individual configurations.

5.1.2 *Stability on Configuration Level*

In our second part of the first research question, we examine the stability of a workload change based on the Kendall´s Tau correlation in the configuration level. The description of Kendall's Tau in the context of stability is described in Section 4.2.2 and 3.5.

RQ1.2: How stable is the relative performance of configurations in the presence of performance changes between different workloads?

Each case study examines the rank correlation of the measured configurations between the two compared workloads. This is intended to visualize the impact of a workload change.

Tar

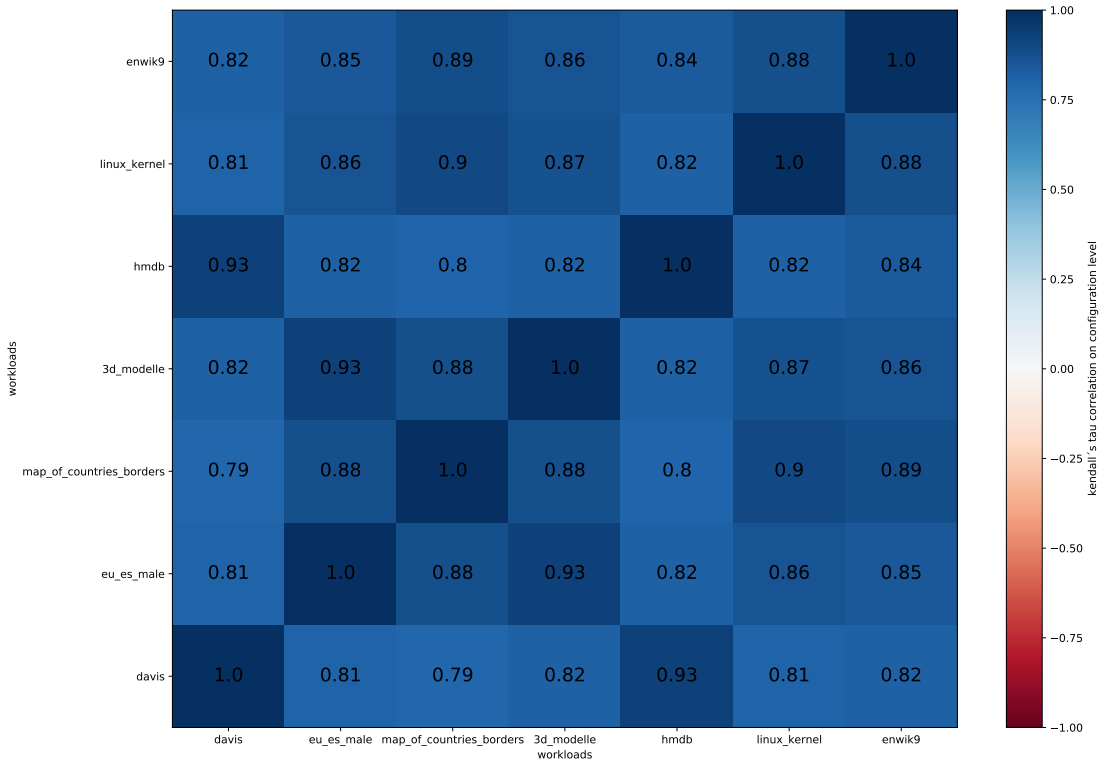

Figure 5.5: Tar_compress stability on configuration level according to Kendall´s Tau correlation. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the Kendall´s Tau correlation value. In addition, these values are displayed in the matrix for the compared workloads.

The results of the case study Tar are represented by Figure 5.5. To consider all workloads, we construct an N x N matrix based on all workloads to compare them with each other. The workloads are all mapped on x-axis and y-axis. The correlations of the two compared workloads are located in the respective section of the matrix. The color scale describes the values from 0 to -1 by a more intense red, whereas the values from 0 to 1 are marked in an increasing blue. Thus, we generally get a fairly high Kendall's Tau correlation of about 0.85, with a maximum of 0.93 and a minimum of 0.79. Furthermore, it is clear that the workloads Linux_Kernel, Enwik9 and Map_of_countries_borders, as well as 3D_models and Eu_es_male and the pair Davis and Hmdb show a higher correlation than with the other workloads. Additionally, the comparisons with Davis or Hmdb perform the worst.
Due to the rather high base correlation for all comparisons, we can say that the order of performance-intensive configurations does not change for the case study Tar. Thus, in terms of performance, the configurations are stable against different workloads. However, we can look more closely at the minimal changes in the Correlations to make further conclusions about stability in the context of the workload. In doing so, we can build on the initial results of RQ1.1 and take the compressibility factor into account. Thus, the just mentioned groupings with increased correlation have the same compressibility level. Thus, for example,

the maximum correlation values arise when we compare the two difficult or medium compressibility workloads with each other. Furthermore, we can see that the strength of the difference in compressibility affects Kendall's Tau. Thus, we obtain the lowest values when we compare light and heavy compressible workloads with each other and slightly increase the correlation when we take medium instead of light compressible workloads. Thus, we find useful insights in this case study, but the different workloads only have a very small effect on the stability of the performance.

Clasp

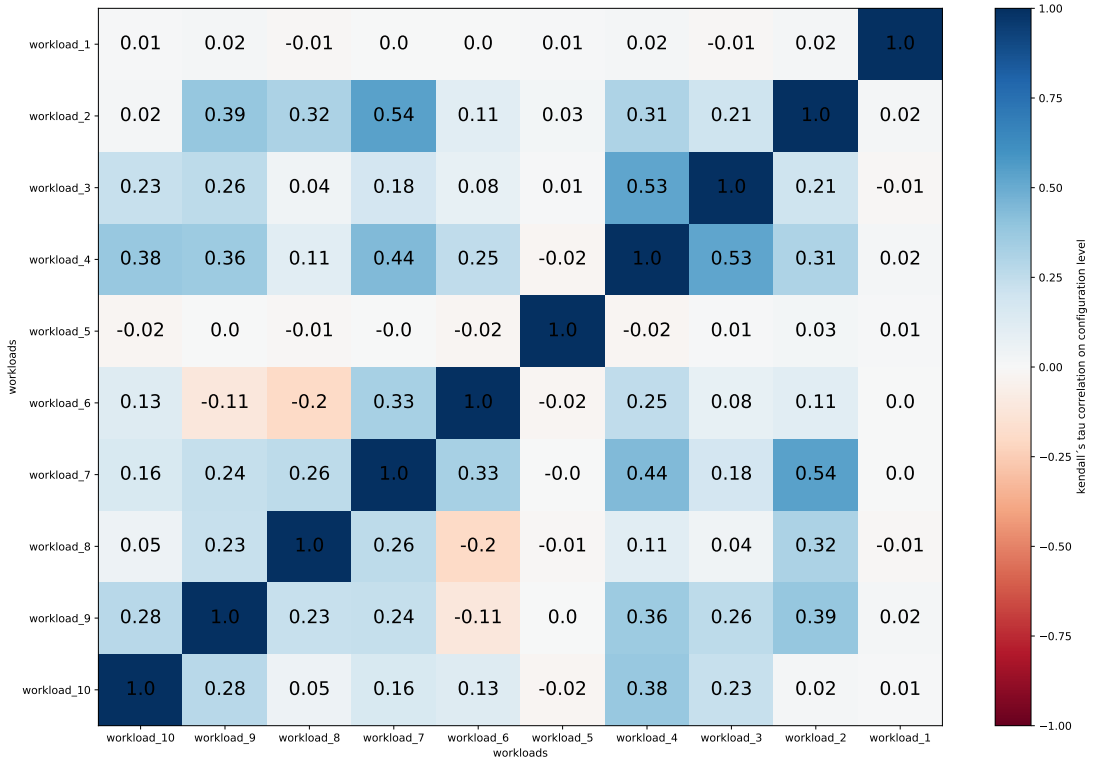

Figure 5.6: Clasp stability on configuration level according to Kendall´s Tau correlation. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the Kendall´s Tau correlation value. In addition, these values are displayed in the matrix for the compared workloads.

Figure 5.6 summarizes the Kendall's Tau correlations of the case study Clasp. When comparing the different workloads, we get results in the range of -0.2 and 0.54. Whereas the general average of the case study is 0.14. This case study, in contrast to the previous ones, also has negative correlation values. We find these in nine of 45 workload comparisons. Furthermore, the range of negative values is lower than the range of positive correlations. The lowest value of 0.2 occurs in the workload workload_6 with workload_8. Whereas the two highest values above 0.5 occur with workloads workload_2, workload_7 and workload_3, workload_4. The workloads workload_1 and workload_5 stand out by

their correlation around the value 0 for all comparators.
Analyzing the presented results in more detail, we can draw further conclusions about the performance impact of workloads. As already seen in RQ1.1 of CLASP, many workloads have their more intensive performance times in different configuration parts. This results in a low rank correlation between workloads, since the order of configurations differs in the context of performance. Due to the weak Kendall's Tau correlation, we can say that the performance in this case study is not stable in terms of workload switching. Taking a closer look at the different workloads, the workloads WORKLOAD_1 and WORKLOAD_5 stand out. Their constant correlation across all workloads reveals their similar behavior. This peculiarity is the cause of the same domain with additional same problem class. Thus, they behave identically at the different workloads except for minimal deviations of 0.03. Furthermore, we find an increased stability with a correlation of more than 0.2 when considering a large part of the SAT workloads. For example, in the comparisons with WORKLOAD_2 we find the values 0.39, 0.32, 0.54 and 0.31, which lead to different SAT workloads, respectively. In contrast, the values of the UNSAT workloads are lower. In the case of CLASP, our observations of stability result in finding differences through the diverse workloads here. The stability is consistently very low and thus does not ensure the same behavior of the configurations with different workloads.

Overall

In the case of RQ1.2, we can summarize that we cannot clearly detect an impact of the workloads on the stability in the configuration layer. In the case study TAR, we did not find any major effects on stability in the context of performance. The same configurations ensured high run times in all workloads. The case study CLASP, on the other hand, showed a completely different behavior. There, the different workloads caused a non-stable behavior in the configurations. The reason for this is that several workloads in different areas have their performance-intensive configurations. Thus, the ranking of the configurations changes to a large extent, which means that Kendall's Tau can no longer establish a clear correlation. The findings of the particular characteristics of a special factor in a system can nevertheless be applied in part to stability. Thus we find a stronger correlation with different workloads of the same factor and thereby a stronger stability. We can observe this especially in the case study TAR. Furthermore, workloads with different factors also differ in terms of stability. Thus, in the area of configurations, a software-specific factor plays a special role in the influence of different workloads.

5.1.3 *Performance Changes on Option Level*

In our second research question, we examine the impact of a workload change based on the performance changes in the option level.

RQ2.1: How frequent and how strong are changes of performance influences of individual configuration options and interactions between different workloads?

In each case study, we examine the absolute impact of the individual configuration options and also use the relative impact to highlight performance changes at the option level. To do this, we use the metric of option-level performance changes described in Section 4.3.
Due to the influence of the different terms, we cannot draw any direct conclusions about the overall performance of the configurations. However, the number and strength of the changing options give us a good estimate of whether workloads have a relevant influence on the system's performance.

Tar

For the case study TAR, we present in Figure 5.7 the influences of the different configuration options and combinations of them from the performance-influence models. All terms are shown on the x-axis and the different workloads on the y-axis. The different color scale shows on the one hand in blue influences that improve the overall run time when activated. On the other hand, run time extensions caused by the respective term are shown in red. In this case study, the number of terms is 163 pieces.
When evaluating the results, we find that the terms increase the performance to a large extent and the range of the performance change of the individual terms is much higher than those that improve the performance. Furthermore, the type of change in performance remains the same in all workloads. Thus, we consistently find performance degradation in all workloads once it occurs in the term. However, we identify differences in the strength of the impact on performance. Thus, we identify certain membership classes in the workloads. The workloads LINUX_KERNEL and MAP_OF_COUNTRIES_BORDERS, as well as 3D_MODELS and EU_ES_MALE and the pair DAVIS and HMDB have a similar change in terms. Thus, we find the smallest effects on the configurations options and combinations for the workloads LINUX_KERNEL and MAP_OF_COUNTRIES_BORDERS. In contrast, the strongest impacts on performance are for the workloads 3D_MODELS and EU_ES_MALE.

Looking more closely at the influences from Figure 5.7, we find that the impact of the different workloads only affects the impact of the same configuration options or combinations of them. Thus, the terms of the system are to a certain extent independent of the workloads, since they do not completely change their behavior due to the workload change. However, one cannot speak of no effects in TAR either. The different workloads provide stronger and weaker influence bounds in the same terms. Thus, there seem to be terms that execute workload specific code, so we see the dependency of the configuration options versus the workloads based on the performance impact. When we put these results in context with the previous results, we also see a grouping of workloads based on the compression factor. The lightly compressible workloads have only a small increase in impact in the prominent terms. In contrast, the medium and heavy compressible workloads stand out clearly. Thus they provide for a very high influence of the known terms. Due to their large amount of uncompressible data, the hard-to-compress workloads have less influence than the medium workloads.

We look at the exact number of performance changes to the configuration options in Figure 5.8. As in RQ1.1, the representation of the results takes place in an N x N matrix

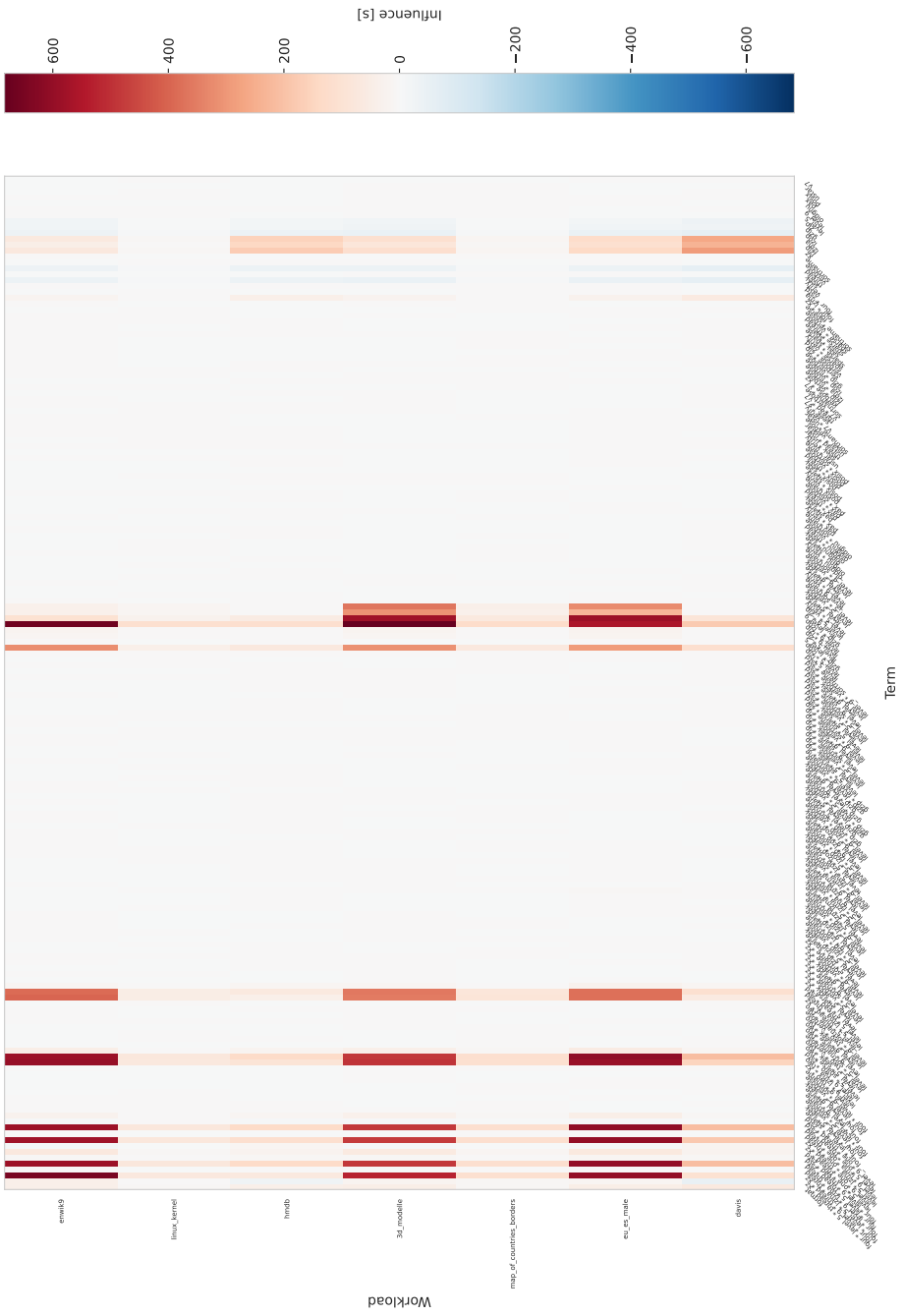

Figure 5.7: Tar_compress performance configuration option and interaction comparison according to all workloads. All configuration options and combinations of them are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective term and workload.

containing all percentage results of the affected terms. The color intensity emphasizes the results compared to the other comparisons. In Figure 5.8 we see that the performance changes between the different workloads affect a small fraction of at most 5.51% of the terms. Furthermore, there are also many workload comparisons that even have no performance changes. Here we can see the same comparison groupings as in the previous figure. If we look at these comparison groups of the workloads, we detected that each comparisons of the two groups Linux_Kernel, Map_of_countries_borders, Enwik9 and Davis, Hmdb have an increased performance change between 4.78% and 5.51%. Thus, we find a similar performance change for each comparison of different groupings. In total, the average number of performance changes between all workloads for the case study Tar is about 2% in the option level.

Figure 5.8: Tar_compress relevant performance changes on option level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

By analyzing the described findings from Figure 5.8 we can draw the following conclusions. We can attribute the very small performance change share of under 6% to the cause of the same performance influence type of terms. So only in a small portion of the terms a too large performance difference arises, so that this ends up in a performance change. Nevertheless, in the case study Tar we find further insights into the impact of different workloads on the performance of the individual configuration options. When looking at compressibility, we discover interesting distributions that give us further points

of workload diversity. When we compare workloads of the same compressibility, we do not get a performance change in any of them. For example, for the easily compressible workloads LINUX_KERNEL, MAP_OF_COUNTRIES_BORDERS and ENWIK9 all compare to each other with a resulting performance change of 0%. Both medium and heavy workloads have the same behavior. Furthermore, the difference in compression is reflected by the changes in performance. Thus, we get the highest changes when comparing light to heavy workloads.

Clasp

First, we again look at the individual performance influences of the individual configuration options and combinations of them in Figure 5.9. We look at whether the same influences run through all workloads. Through the performance-influence models of the workloads, a total of 45 different terms are considered in the case study CLASP.

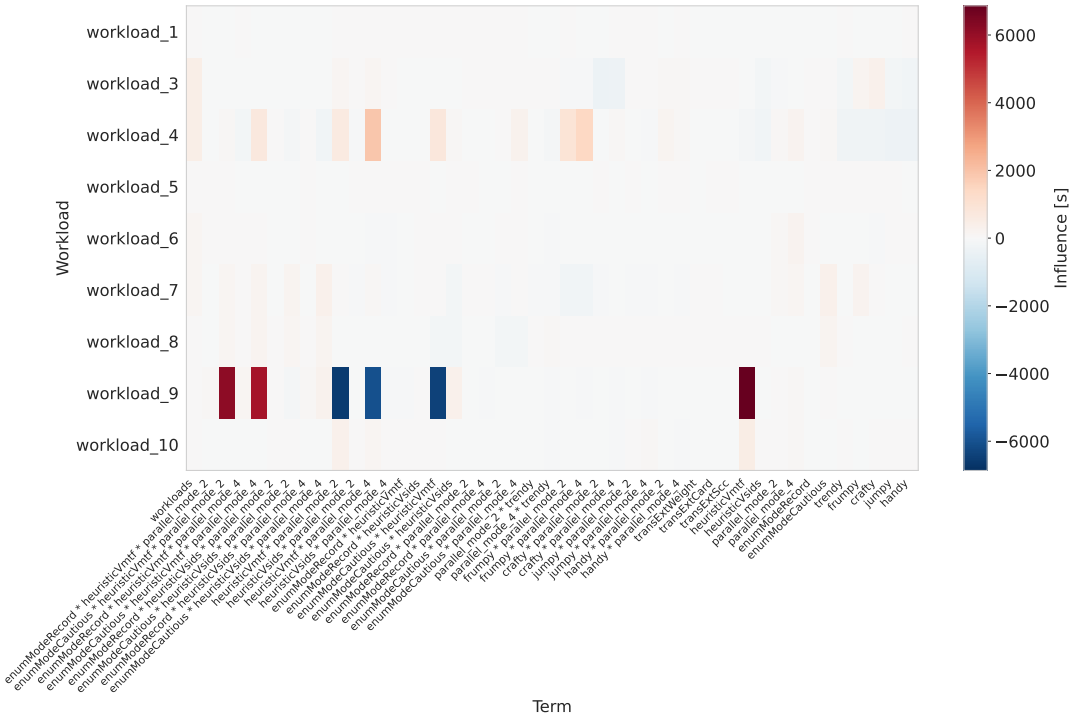

Figure 5.9: Clasp performance configuration option and interaction comparison according to all workloads. All configuration options and combinations of them are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective term and workload.

In Figure 5.9 we consider all workloads except for WORKLOAD_2, since this one disturbs the readability of the graph by too extreme values as in RQ1.1. The graph with all workloads is included in the appendix in Appendix A.4. Besides the missing workload, it is noticeable that we can show higher dynamics in the different terms than in the previous case

studies. Thus, in addition to high performance degradations, we also see high performance improvements associated with different terms. In addition to the workload WORKLOAD_2, the workload WORKLOAD_9 has several terms that show a high impact on performance. We notice that the same configuration options and combinations do not necessarily have the same performance impact in all workloads. For example, it happens that, among others, the workloads WORKLOAD_4 and WORKLOAD_9 in the term HEURISTICVMTF * PARALLEL_MODE_4 on the one hand cause a performance improvement and on the other hand a degradation. The workloads WORKLOAD_1 and WORKLOAD_5 show very little change in all terms, so we cannot distinguish them in the figure.
To analyze the data, we can note the following. As in the previous case studies, the workloads are also prominent in the configuration options, which also achieve high performance values in the configuration level. However, it is noticeable here that the change of the workload does not only cause a stronger or heavier impact of the same term, but switch terms. This gives us workload dependent configuration options and combinations that only behave this way under these circumstances. Another reason for the workload dependency is the different effects of the same terms on the workloads. For example, the term HEURISTICVMTF * PARALLEL_MODE_4 between the different workloads there are run time improvements, degradation's and no impact. In contrast, there are also workloads such as WORKLOAD_1 and WORKLOAD_5 that were not influenced by the workload in any term.

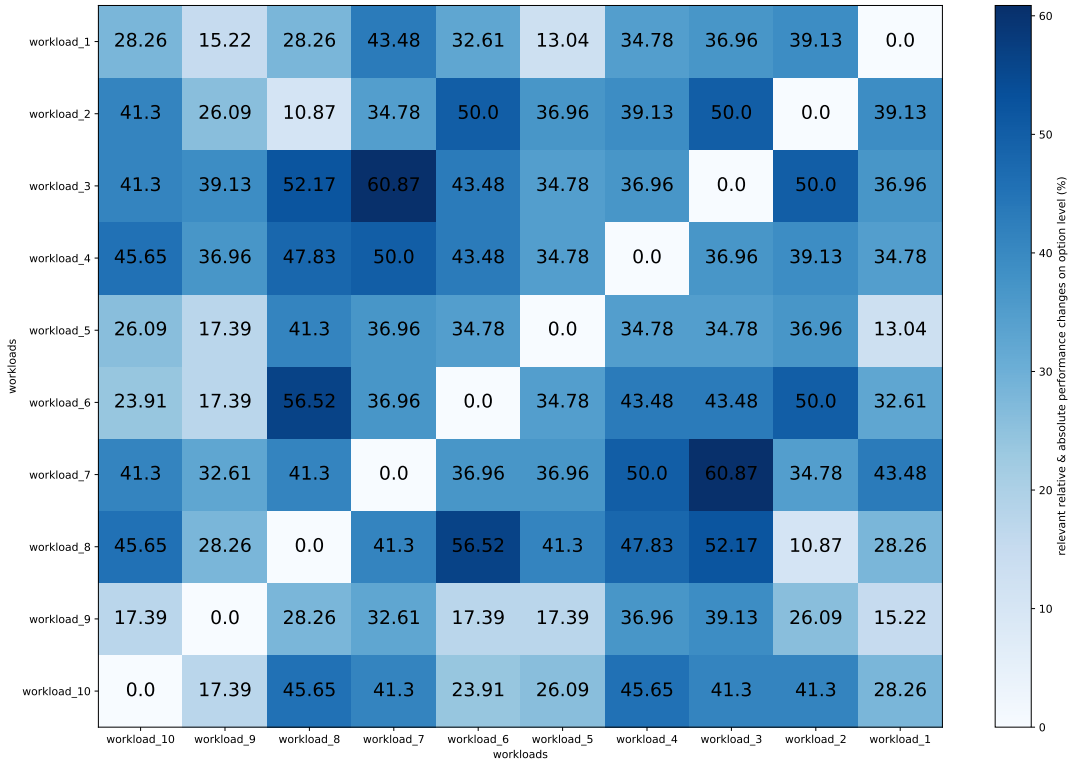

Figure 5.10: Clasp relevant performance changes on option level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

In the next step, we look at the percentage of performance changes in Figure 5.10. We get values between 10.87% and 60.87%. We find the minimum at workloads WORKLOAD_2 and WORKLOAD_8, whereas the maximum is at workload WORKLOAD_3 and WORKLOAD_7. Furthermore, the number of performance changes are very different, so that no further patterns or clusters can be identified. The average number of performance changes between all workloads for the case study CLASP is about 35%.
Due to the relatively mixed results from Figure 5.10, we can only gain a few insights. First, there is a strong part of configuration options and combinations that behave independently or only very weakly depending on the workloads. Thus, we find only a small part in the different terms that adapts its behavior more strongly to the workload. In the case of the workloads WORKLOAD_1 and WORKLOAD_5, which we could already observe in RQ1.1, we also find the same behavior in the option level. Thus, the same domain and problem class ensures similar behavior across configuration and configurations option for a lower number of performance changes. By changing the performance-intensive terms, we obtain a higher average number of performance change than in the case study TAR. However, this percentage value cannot keep up with the configurations level of both case studies.

Overall

In summary, we were able to determine the effects of different workloads on the influences of the individual configuration options and combinations of them. However, we must distinguish that on the one hand, as in the case study CLASP , there are systems, which is affected by a larger part performance changes. This is justified by the fact that the workload does not only cause a performance change for the same terms. On the other hand, the case study TAR describes the systems in which the same terms cause a performance change for all workloads. Thus, the number of performance changes is not necessarily a good reference for whether different workloads affect the run time of the configuration options. However, by looking further at the case studies using the performance-influence models, we were able to determine that they can affect the performance of the configuration options. If we accumulate both case studies we get an average number of performance changes of around 19%. However, individually they are 2% for TAR and 35% for CLASP, so we can still see a relatively large difference between them.
Furthermore, in the case study TAR and in the smaller framework of CLASP, we were able to take another look at the determining factors of the case studies. This allowed us to extend the conclusions of the configurations level to the performance changes of the option level. This is because we find the same groupings of values here as well. Thus, workloads with the same factor also differ in the configurations options to a very small extent and the greater the difference of the factor, the performance changes will also be higher.

5.1.4 *Stability on Option Level*

In our second part of the second research question, we examine the stability of a workload change based on the Kendall´s Tau correlation in the option level. The description of

Kendall's tau in the context of stability is described in Section 4.2.2 and 3.5.

RQ2.2: How stable is the influence of configuration options and interactions in the presence of performance changes between different workloads?

To answer the research question, in all case studies we consider the rank correlations of the configuration options and combinations of them that appear in the created performance-influence models. This is to highlight the differences of different workloads on the terms. The required rank lists are determined by the influences of the terms (see Section 4.2.2).

Tar

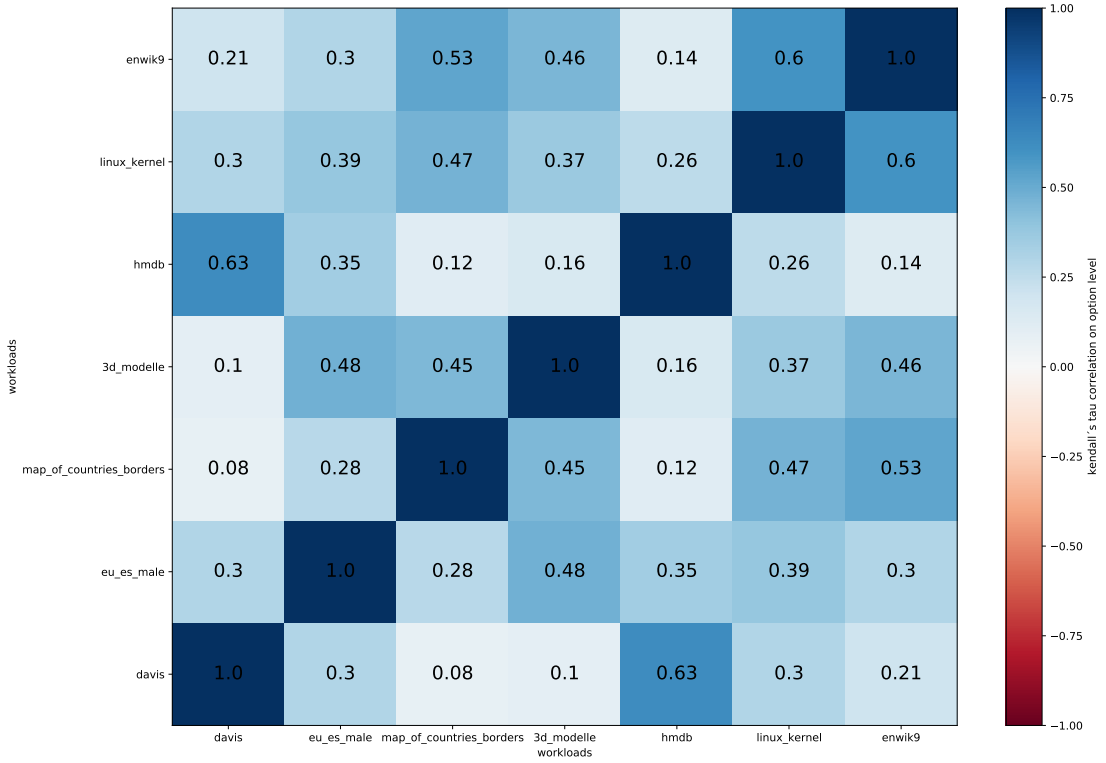

Figure 5.11: Tar_compress stability on option level according to Kendall´s Tau correlation. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the Kendall´s Tau correlation value. In addition, these values are displayed in the matrix for the compared workloads.

The results of the case study TAR are represented by Figure 5.11. To better illustrate the results of Kendall's Tau correlation, the following figures describe a matrix of all the examined workloads. They are all mapped on x-axis and y-axis. The correlations of the two compared workloads are located in the respective section of the matrix. The color scale describes the values from 0 to -1 by a stronger red, whereas the values from 0 to 1 are marked in an increasing blue. So we use the same matrix and notation as RQ1.2.

In general, we find a high correlation between the different workloads. Thus, we find correlations between 0.08 and 0.63. The highest correlations are in the comparisons of the workloads Davis and Hmdb, Linux_Kernel and Enwik9, and Linux_Kernel and Map_of_countries_borders. We find the lowest values for Map_of_countries_borders and Davis, and Map_of_countries_borders and Hmdb.
The consistently positive Kendall's Tau correlations reinforce the claim that the same terms remain the performance-intensive terms in all workloads. Thus, we find terms that cause run time degradation in all workloads in the upper ranges of the rank correlation. Furthermore, we can conclude very strong similarities to the stability in the configuration level. There we find in the same comparisons, both the minimum and the maximum Kendall's Tau correlations. Based on this, we find here even more clearly the individual workload groupings based on compression. The workloads with the same compression level also provide a stronger correlation in the configuration options. Whereas differently compressible workloads have a lower correlation and therefore stability. In addition to the comparison with the configuration level, we can compare the stability results with those of the performance changes on the same level. Thus, we find an inverse behavior for Tar compared to the two research parts. Basically, this is also a logical conclusion, because if we find a lower stability in the configuration options, this automatically causes performance changes in the terms.

Clasp

Figure 5.12 contains the results of the Kendall's Tau correlation of the case study Clasp. When comparing the different workloads, they are in the range of -0.24 and 0.45. The average is 0.04 and is thus almost 0, which means that the correlation cannot be used to make a statement about the ranking. We also find negative correlation values in the option level only in the case study Clasp. We could make the same observation before in the configuration level as well. Furthermore, we see increased correlation in the comparisons of workload_6 and workload_9, as well as workload_2 and workload_8 and workload_3 and workload_5. In addition to that, we see an increased negative correlation for workloads workload_3 and workload_9, and workload_5 and workload_8 and workload_3 and workload_6.
Analyzing the presented results in more detail, we can draw further conclusions about the performance impact of workloads. As already seen in RQ2.1 of Clasp, many workloads have their more intense performance influences in different configuration options and combinations of them. This results in a low rank correlation between workloads, since the order of terms in the context of performance differs. Due to the low Kendall's Tau correlation in the whole case study, we do not speak of a stable workload alternation here. Thus, we cannot transfer the ranking of terms. Conversely, this means that the different workloads play for a strong variation in the order of configurations options in the context of performance. Thus, even by looking at stability from a different angle, we see that the workloads in the case study Clasp have an impact on performance.

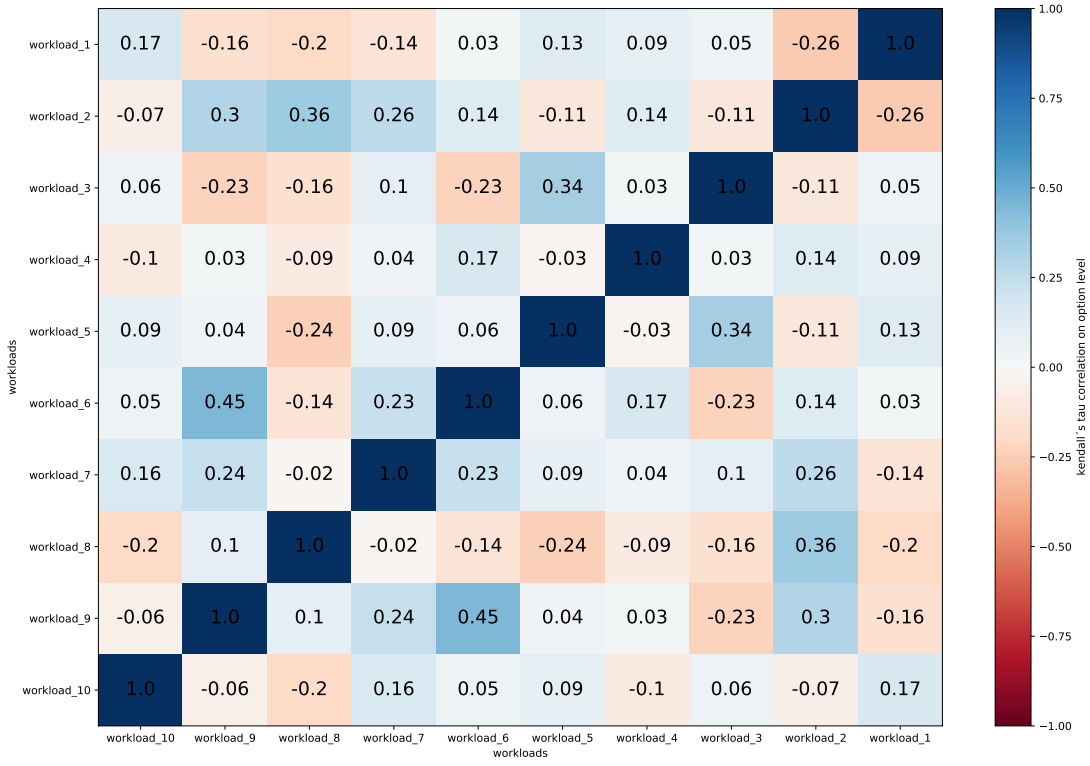

Figure 5.12: Clasp stability on option level according to Kendall´s Tau correlation. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the Kendall´s Tau correlation value. In addition, these values are displayed in the matrix for the compared workloads.

Overall

For RQ2.2, we can summarize that the influence of the workloads on the stability of the configuration options plays a more definite role than in the configuration level. Thus, while we do not have unstable behavior due to the workloads in all case studies. However, in each case study we find that the different workloads cause different stability in general. In the case study Tar, we observe exactly this behavior. There we discover correlation values that evolve from very stable values due to a different workload to a completely different value and thus very unstable behavior in the ranking. Such jumps are the effects of different workloads. For the case study Clasp, we get even clearer correlations for this. The values are distributed around the value 0, so that we can no longer speak of a clear ranking. The reason for this are the different terms, which provide a strong influence due to different workloads. This was already discussed in RQ1.1. This is the reason for the unstable behavior in the option level.
If we finally look at the important factor of workload diversity, we get similar results as in the configuration level. Thus, we find a stronger correlation by the same factor here as well. Therefore, in both levels, a large difference in this factor results in more unstable rankings. This can be seen as another reason for the influence of different workloads.

5.2 DISCUSSION

In the following section, we still want to discuss specifics that we found during our research and answering of the research questions.
To begin, we want to address the diversity of the workloads, as these were an important component of this work. We attempted to cover a broad workload spectrum through as much diversity as possible, thereby representing a wide variety of influences. In this work we compared workloads that differed in several aspects. At the same time, however, we made sure that it was very difficult to reduce the observed differences to a single characteristic. In the case study TAR, on the other hand, we also considered workloads that differed from other workloads in only one aspect, which allowed us to make more well-founded statements about the different influences. Thus, in further research of the work, I would continue to insist on the diversity of workloads, but pad it with additional workloads to look at the differences in a somewhat more fine-grained way.
Furthermore, the importance of a software dependent factor in workload selection became clear. Thus, using this distinction, it was possible to divide the workloads into groups that showed a connection throughout all of the research results. Thus, workloads of the same group provide identical behavior towards other groups. Among other things, this allowed us to classify workloads that are more differentiated and thus cause a greater impact on performance. Thus, it would make sense to adjust the workload selection to this factor in order to obtain unambiguous results.
Finally, we want to discuss the measurements of TAR_EXTRACT, because although the measurements are from the same case study, the results of TAR_EXTRACT and TAR_COMPRESS are very different. The results of TAR_EXTRACT can be seen in the appendix under Appendix A.3. Thus, although the different measurements describe the same software system, the different tasks occupy two completely different functional areas. This also changes the distribution of the performance values and the intensity of the individual configurations and configuration options. As a result, other configurations and configuration options describe the higher performance influences of the measurements.

6 THREATS TO VALIDITY

In this section, we discuss the threats to internal and external validity.

6.1 INTERNAL VALIDITY

This section addresses the limitations and potential causes of errors in our measurements. Therefore, we list limitations in our experimental setup or overall research that may lead to incorrect or inaccurate results. For each of these reasons, we describe the steps we have taken to mitigate or eliminate the threat to the validity of our results.
When examining performance measurements, we must ensure that they are not affected by random fluctuations between different measurements. Otherwise, we will not get accurate measurement data, because individual configurations will get a different run time than usual. To prevent this behavior, we measure all configurations five times and form a relative standard deviation from all values. If this is above 10% of the measurements, the configuration is measured again until the value is below the border. The measurements are additionally performed on a cluster of computers. Thus, the individual iterations are partially calculated in parallel on different independent workstations. Thus, we obtain five independent measurement results, which can be compared.
To avoid the influence of software running in the background, we use a scheduling tool called Slurm[1]. This takes care of the scheduling and execution of the individual workloads. It takes care that no jobs are executed on the same node at the same time. As a further point, the cluster has only a lightweight operating system without a GUI, which has no significant influence on the measurements. Therefore the background software is kept minimal.
Another potential performance impact could be the access to the required data. For example, the data could be stored centralized in the cluster and all nodes would have to access the same point in the system. This could allow certain machines to get to the requested data faster due to different cache behavior. To avoid this, we ensure that each job has the required data as a separate copy on the executing node before the measurements. Since all nodes have the same underlying data structure, each node has its own data that requires the same access time.
In some of the case studies, numeric configuration options are used in addition to binary configuration options. This provides a different representation in the performance-influence models. Here, all values of a numeric configurations option are represented as one feature. As a result, we lose the consideration of the individual values. To combat this, we have converted all numeric configurations options into binary configurations options. This way we are able to look at each numeric value individually.
Furthermore, the operationalization and evaluation of the measurement results can lead to potential errors. We use different metrics to answer our research questions. If we were to choose the parameters of these metrics differently, we would arrive at different results,

1 https://slurm.schedmd.com/documentation.html - last visited on 17.11.2021

leading to wrong conclusions. Therefore, we use, among others, the standard deviation or the influence of both workloads as parameters to consider the different workloads equally and to compensate for high differences if necessary. In addition, the mean values of the performance provide an estimate that, depending on the compared workloads, ignores smaller changes that we can evaluate as noise in this case.

6.2 EXTERNAL VALIDITY

To support the generalizability of our results, we considered two different case studies. Two different application areas, a solver and a compression tool, were investigated. Thereby, the compressing and the extracting were measured. In addition to the different case studies, a diverse number of configuration options were considered, which have a different effect on the run time to be investigated. Another important point is the diversity of the selected workloads. They ensure that the most diverse domains, difficulties and sizes of workloads were considered. Thus, first major steps towards generalizability were taken by the broad selection and investigation of features and workloads, but further case studies from different application domains have to be investigated to make a generalized statement of the results. This was not possible in the given time span of the bachelor thesis.

7

CONCLUSION AND FUTURE WORK

CONCLUSION

In this work, we consider the impact of different workloads on the performance of highly configurable systems. More specifically, we examine the effects that occur when the workload is changed. In addition to the number of performance changes, we also study the stability of the system. To answer the research questions related to the overall topic, we consider two case studies - Tar and Clasp. Before we could measure the case studies and learn the respective models, we analyzed the different systems to find potential workload candidates and to perform a feature selection. Furthermore, the measurement data of the case studies were checked for multicollinearity and, if necessary, terms causing it were removed. Besides the basic performance values, we use these performance-influence model to look at individual configuration options and interactions thereof. After the preparation and measurements, the case studies are conducted and their results are used to answer the research questions.
First, we consider the configuration level and thus use the performance values of the individual configurations. We first examine the relevant performance fraction that changes between different workloads. By analyzing the case studies, we were able to determine that different workloads can trigger a performance change of about 78%. This always depends on the differences between the workloads. However, we were able to detect a small performance change even with almost identical workloads. As a second point of view, we observe the stability of the configuration order, which was established on the base of the performance. We cannot give a hard answer to this question, since our case studies show that this depends on the system. Thus, we find a high stability with a Kendall's Tau correlation above 0.5 for systems where the performance-intensive configurations do not change among the different workloads. In the reverse case, the configurations are not stable and get a correlation around 0. We could not find out in this work when a system differs in the performance-intensive configurations.
The second part of the research questions dealt with the option level. Thereby the configurations options and combinations of them of the prepared performance-influence models are treated. When examining the number of relevant performance changes of different workloads, we found that each workload change caused performance changes. Thus, on average, about 19% of configurations options and interactions of them have changed their performance. However, the average varied between 2% and 35%, so we see a connection to the measured system here as well. As in the configurations, we recognize the differences in the distribution of the performance-intensive terms. For example, in one system they stay the same and provide a low number of performance changes and in another system they change depending on the workload and affect the performance change to a greater extent. Second, we analyzed the stability of the configurations options and combinations of them. We found the same behavior of the terms as in the configurations, but in a slightly

weaker manner. Thus, the stability in the option level also depends on the system under investigation.
When we analyzed the data, we were able to divide the results of the workloads by a particular system-specific property in each of the research questions. For example, in one case study, the compressibility of the workloads played a crucial role, so we were able to distinguish how similar the workloads were based on that. The workloads of the same class behaved similarly, so that we were able to detect smaller performance changes and stable behavior. Moreover, a higher difference between the workloads in terms of the property resulted in a higher number of performance changes and a decrease in stability. Thus, we were able to determine the properties of the system as an important subdivision of the workloads, which, based on this, cause a higher impact on performance.

FUTURE WORK

In this chapter, we outline possible further work that emerges from this thesis.
As already indicated by the external validity from Chapter 6, we cannot relate our results to all highly configurable systems. This is due to the fact that only a very limited range of case studies was investigated due to time constraints. A possible further work would be to continue this research. The same configuration settings and research approaches should be investigated in further case studies and application examples. In this way, further conclusions can be drawn about this research and a generalizable result can be reached by evaluating all results together. When selecting further case studies, a focus should be placed on the selection of different domains in order to cover a broad spectrum of software types.
Another similar area of research is illustrated by the underlying paper [17] in the context of this work. Their influence models are used to predict performance values as well as energy consumption. Thus, a further study can deal with the effects of different workloads on the energy consumption of highly configurable systems. The same approach as in this thesis can be used to compare the results of performance and energy consumption and to find commonalities and correlations.
As a last further work we want to describe the investigation of further possible influences on the performance. As already mentioned there is the possibility to compare different software revisions to detect performance changes between two successive revisions (see Chapter 4.2.2). In this way, both workloads and different releases can be examined later to compare the different influences on the run time. In this way, performance changes between two versions can be detected, which, for example, only occur with specific workloads.

A

APPENDIX

A.1 CONTENT OF THE ACCOPANYING USB STICK

The enclosed USB stick contains all necessary data to reproduce the case studies and the results. On the other hand, all measured values and generated plots of the case studies are stored too. Thereby different files, like for example the measured workloads are packed in archives, because otherwise they need too much memory. Furthermore, there is a digital version of the thesis on the storage medium. The following list describes the data and folder structures of the included USB stick:

thesis.pdf
: This thesis as PDF file.

clasp
: Contains all files and plots of case study Clasp.

tar
: Contains all files and plots of case study Tar.

Furthermore, the above mentioned folders of the case studies contain the following files:

deviations.csv
: Deviation results of the case study in CSV format.

measurements.csv
: Messurement results of the case study in CSV format.

models.csv
: Performance-influence model results of the case study in CSV format.

featuremodel.xml
: Feature model with workloads of the case study for SPL Conqueror.

workloadmodel.xml
: Feature model without workloads of the case study for SPL Conqueror.

plots
: Contains all plots of the case study.

workloads.zip
: Compressed file, which contains all measured workloads.

binaries
: Contains all binaries, which are necessary for the measurements.

start.sh
Shell script used to start the measurements for the case studies.

configurations.csv
CSV file containing all measured configurations.

In the case study TAR the .csv files except the configurations.csv are divided by the two different measurements into the folders /compress and /extract.

A.2 ADDITIONAL PLOTS OF THE CASE STUDY TAR_COMPRESS

This section contains additional graphics that illustrate the measurements and results of the case study.

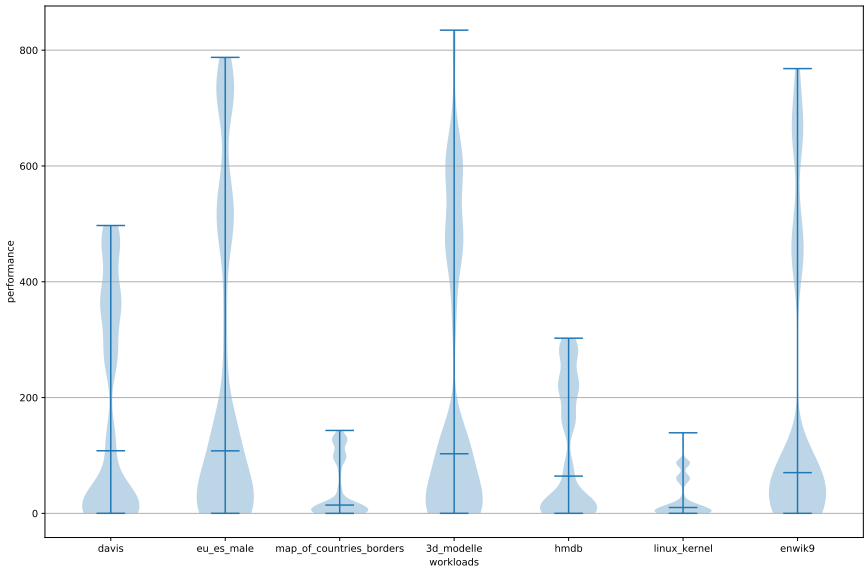

Figure A.1: Tar_compress performance violin comparison with all workloads. All workloads are depicted on the x-axis and the y-axis describes the performance.

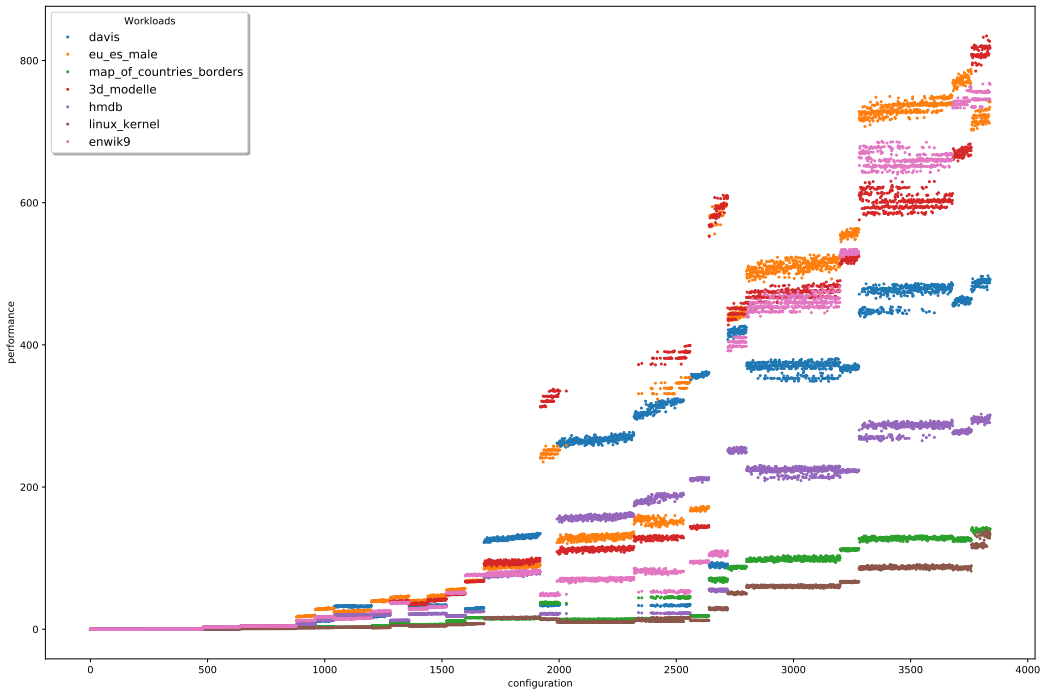

Figure A.2: Tar_compress performance scatter comparison with all workloads. All configurations are depicted on the x-axis and the y-axis describes the performance.

A.3 ADDITIONAL PLOTS OF THE CASE STUDY TAR_EXTRACT

This section contains additional graphics that illustrate the measurements and results of the case study.

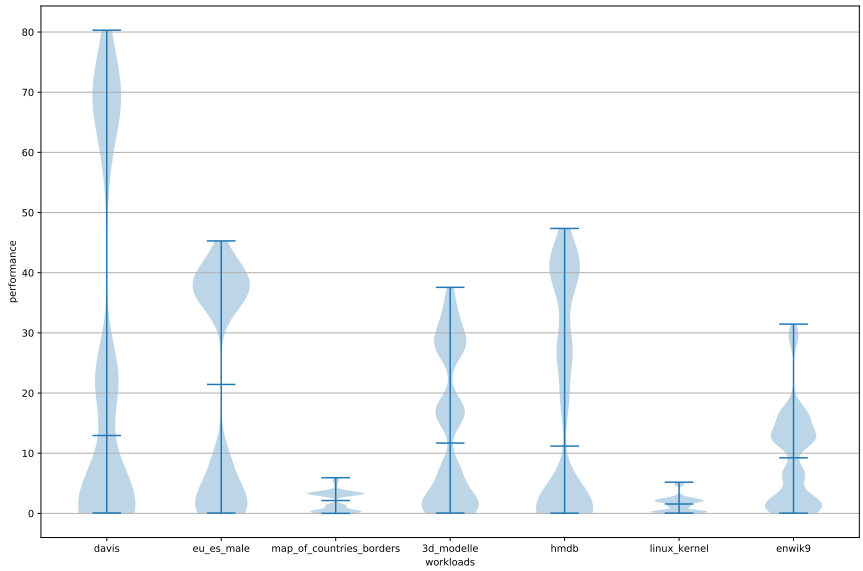

Figure A.3: Tar_extract performance violin comparison with all workloads. All workloads are depicted on the x-axis and the y-axis describes the performance.

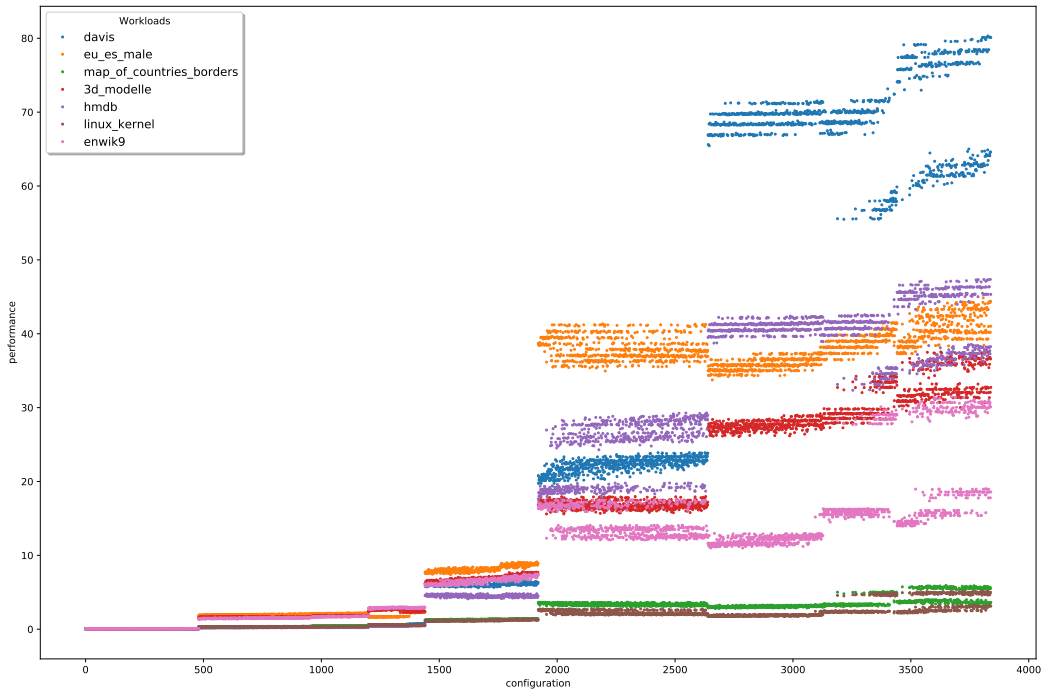

Figure A.4: Tar_extract performance scatter comparison with all workloads. All configurations are depicted on the x-axis and the y-axis describes the performance.

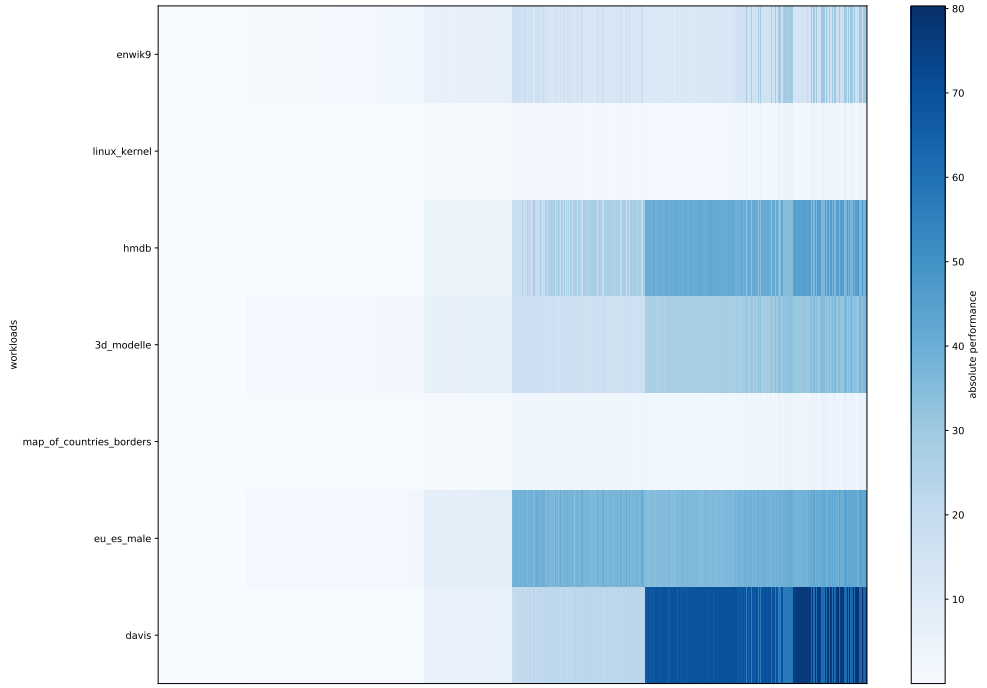

Figure A.5: Tar_extract performance configuration comparison according to all workloads. All configurations are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective configuration and workload.

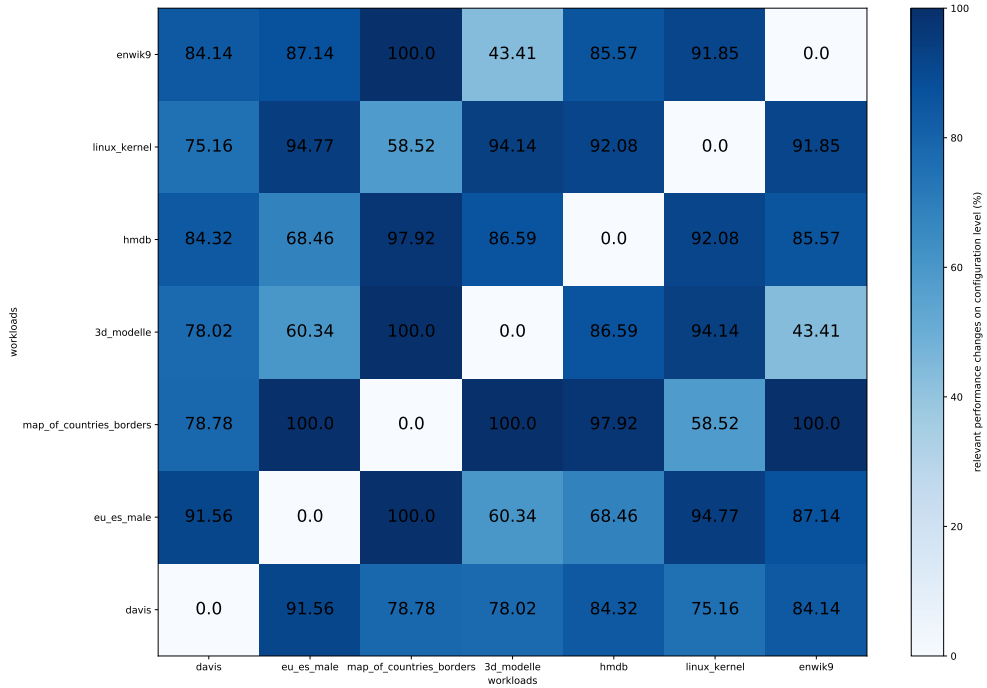

Figure A.6: Tar_extract relevant performance changes on configuration level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

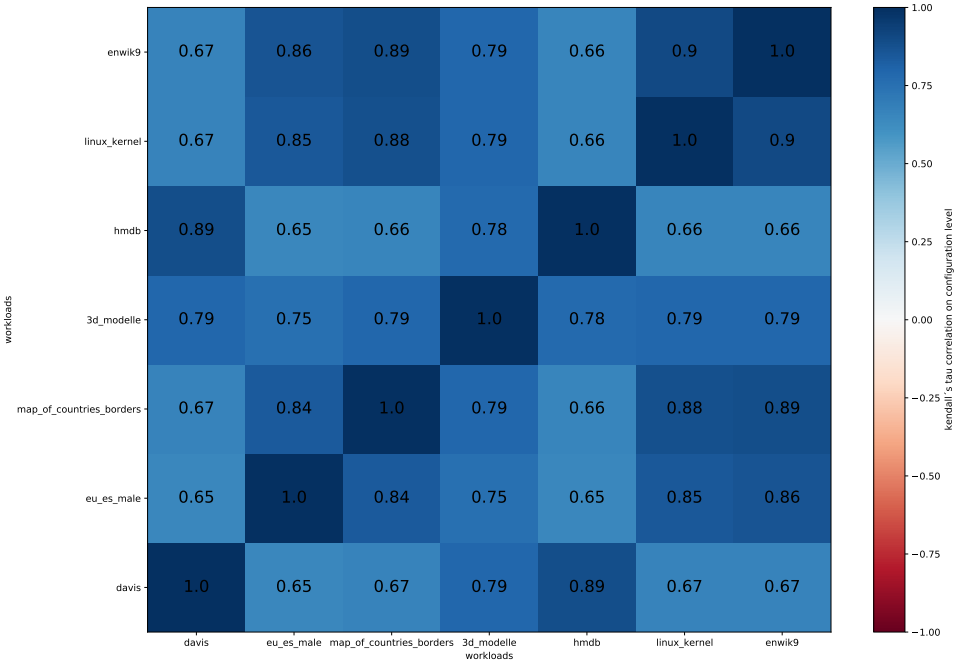

Figure A.7: Tar_extract stability on configuration level according to Kendall´s Tau correlation. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the Kendall´s Tau correlation value. In addition, these values are displayed in the matrix for the compared workloads.

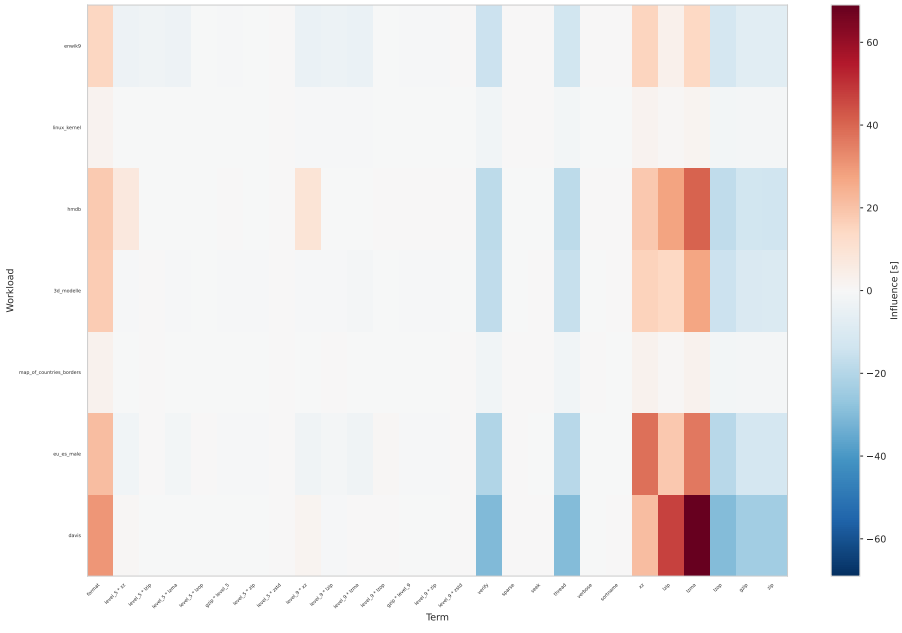

Figure A.8: Tar_extract performance configuration option and interaction comparison according to all workloads. All configuration options and combinations of them are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective term and workload.

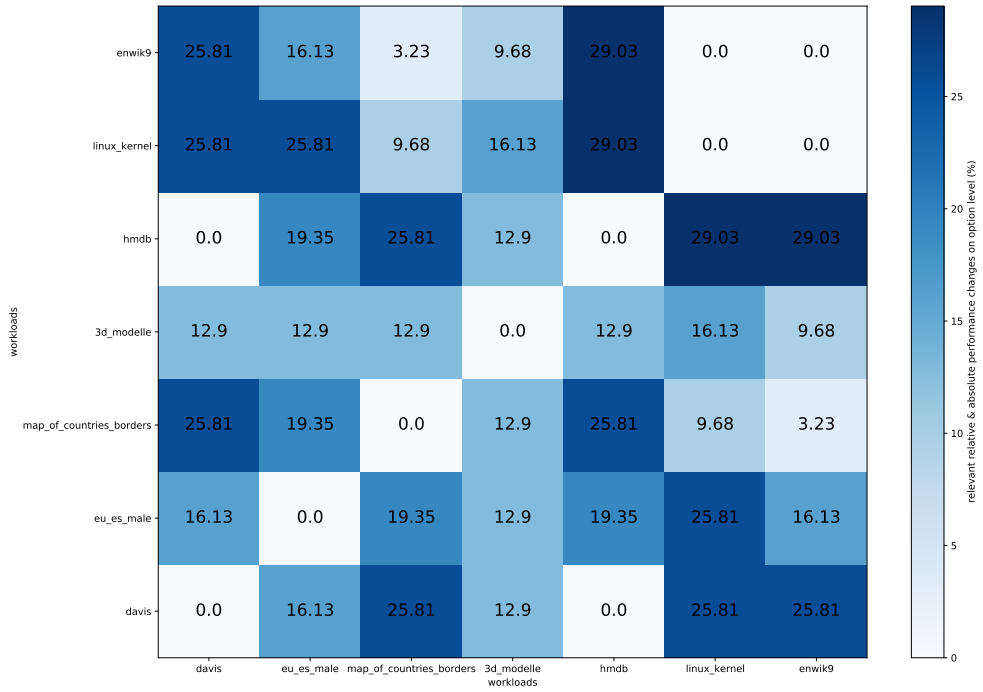

Figure A.9: Tar_extract relevant performance changes on option level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

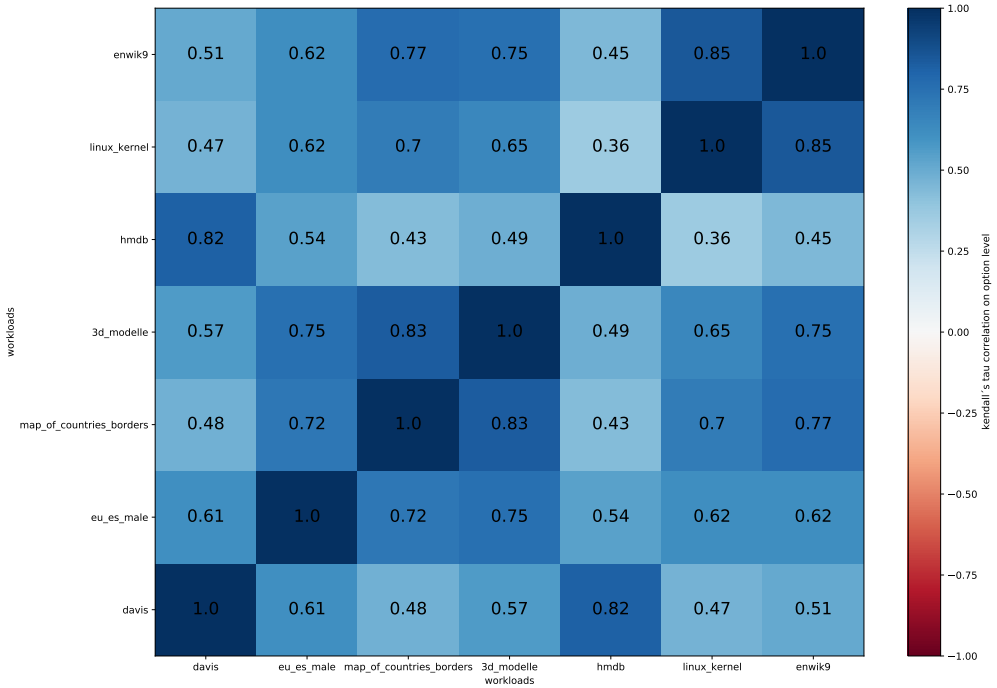

Figure A.10: Tar_extract stability on option level according to Kendall´s Tau correlation. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the Kendall´s Tau correlation value. In addition, these values are displayed in the matrix for the compared workloads.

A.4 ADDITIONAL PLOTS OF THE CASE STUDY CLASP

This section contains additional graphics that illustrate the measurements and results of the case study.

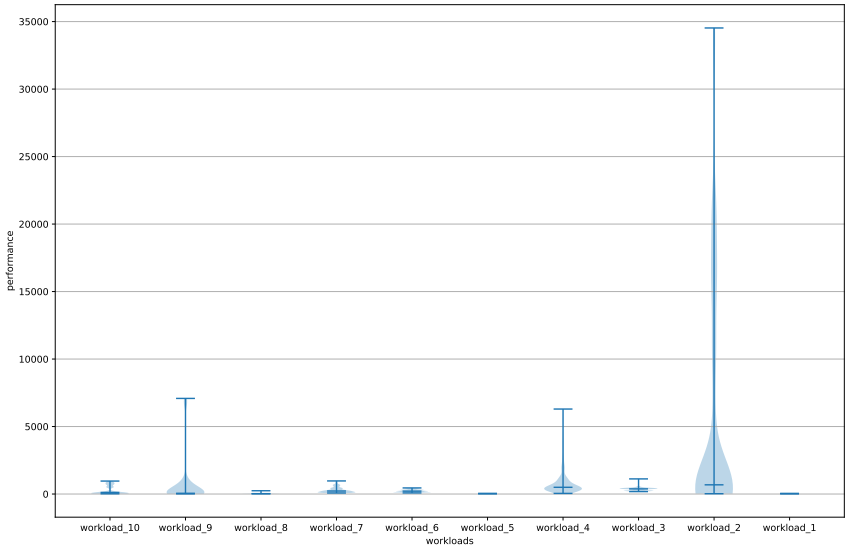

Figure A.11: Clasp performance violin comparison with all workloads. All workloads are depicted on the x-axis and the y-axis describes the performance.

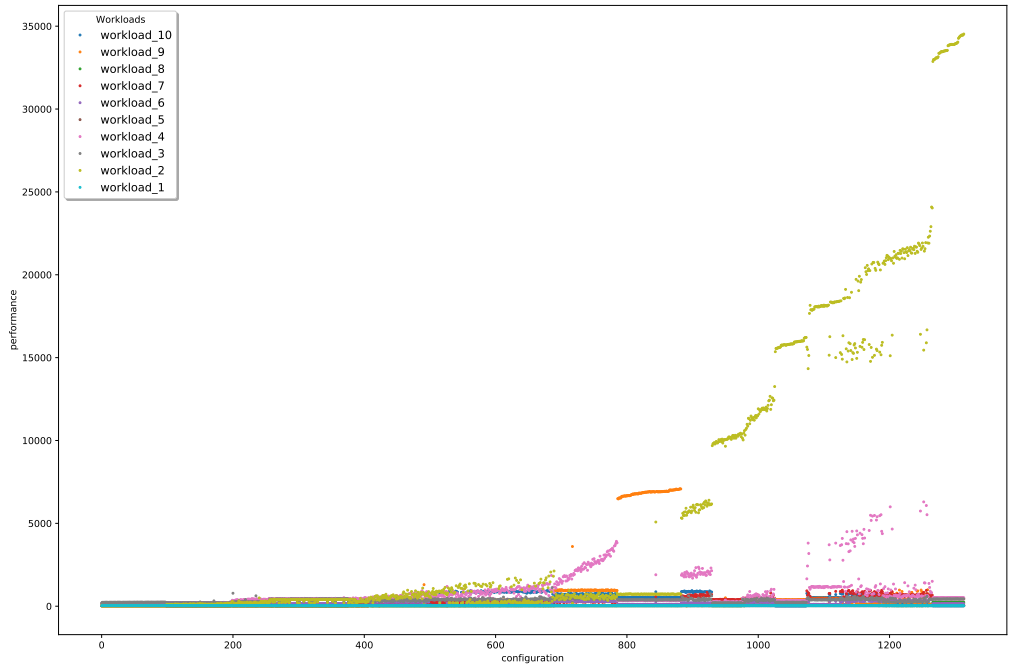

Figure A.12: Clasp performance scatter comparison with all workloads. All configurations are depicted on the x-axis and the y-axis describes the performance.

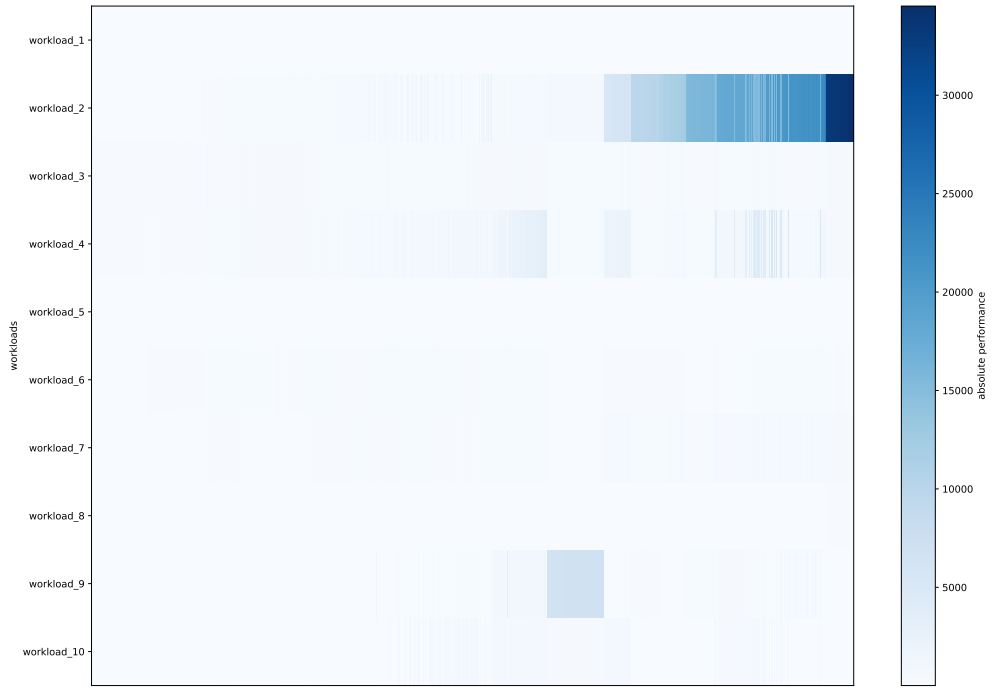

Figure A.13: Clasp performance configuration comparison according to all workloads. All configurations are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective configuration and workload.

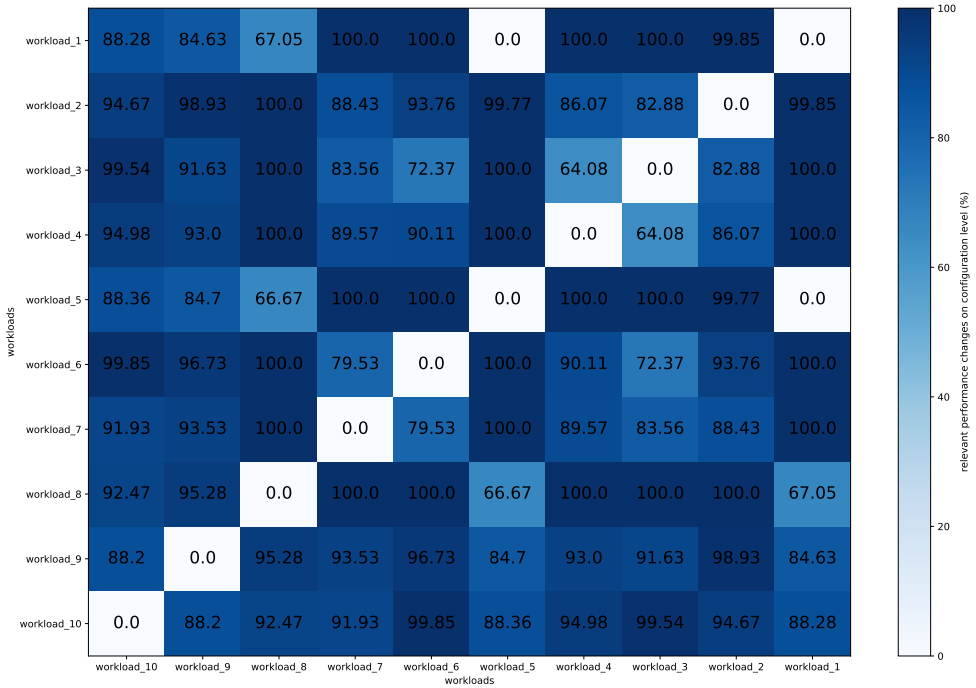

Figure A.14: Clasp relevant performance changes comparison on configuration level according to all workloads. All workloads are depicted on the x-axis and y-axis. The color intensity is used to show the percentage of relevant performance changes. In addition, these values are displayed in the matrix for the compared workloads.

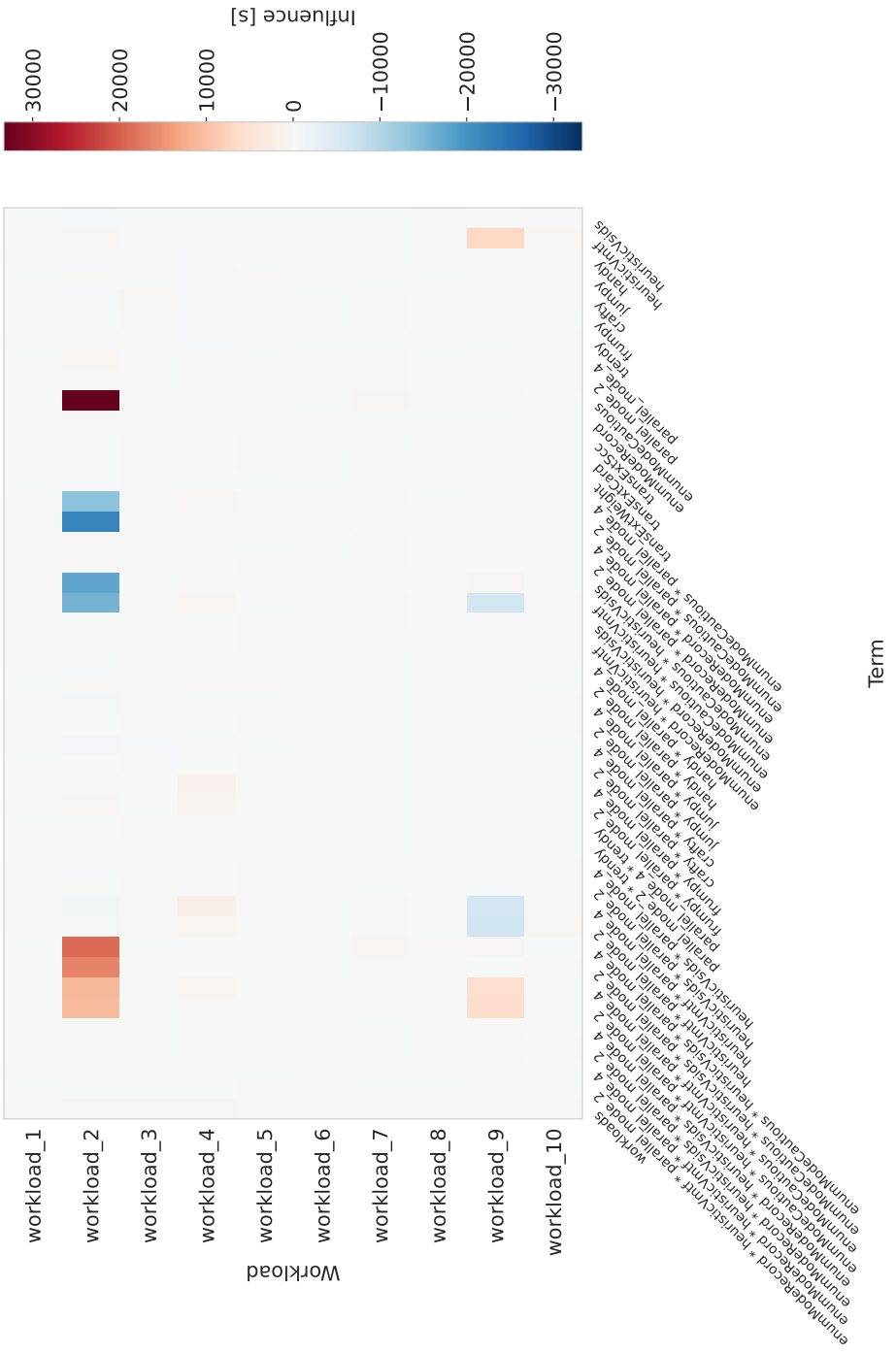

Figure A.15: Clasp performance configuration option and interaction comparison according to all workloads. All configuration options and combinations of them are depicted on the x-axis and the y-axis describes the different workloads. The color intensity is used to show the performance of the respective term and workload.

A.5 VIF ANALYSE OF THE CASE STUDIES

The following tables describe the VIF of the features that have been assigned an influence based on the performance-influence models. Furthermore, the terms have already been checked for multicollinearity and, if necessary, features have been removed. The tables serve to illustrate that the examined performance-influence models are free of perfect multicollinearity. Since TAR_COMPRESS and TAR_EXTRACT have the same VIF analysis results, it is mapped only once below under TAR.

features	VIF factor
Intercept	52.5
xz[T.1]	14.9
lzip[T.1]	15.2
lzma[T.1]	17.7
lzop[T.1]	15.2
gzip[T.1]	15.2
zip[T.1]	13.2
verify[T.1]	8.0
oldgnu[T.1]	7.8
pax[T.1]	7.8
posix[T.1]	7.8
ustar[T.1]	5.7
v7[T.1]	5.7
seek[T.1]	19.2
verbose[T.1]	18.1
sortname[T.1]	19.2
level_5[T.1]	26.3
level_9[T.1]	26.3
level_5[T.1]:sparse[T.1]	9.1
level_5[T.1]:seek[T.1]	16.7
level_5[T.1]:verbose[T.1]	16.7
level_5[T.1]:sortname[T.1]	16.7
level_9[T.1]:sparse[T.1]	9.1
level_9[T.1]:seek[T.1]	16.7
level_9[T.1]:verbose[T.1]	16.7
level_9[T.1]:sortname[T.1]	16.7
level_9[T.1]:two[T.1]	16.7
level_9[T.1]:xz[T.1]	16.0
level_9[T.1]:lzma[T.1]	17.1
lzma[T.1]:two[T.1]	2.9
level_9[T.1]:zstd[T.1]	10.3
level_5[T.1]:two[T.1]	6.9
seek[T.1]:two[T.1]	5.6
sortname[T.1]:two[T.1]	5.6
level_5[T.1]:xz[T.1]	15.7
seek[T.1]:xz[T.1]	10.9
verbose[T.1]:xz[T.1]	10.2
sortname[T.1]:xz[T.1]	10.9
level_5[T.1]:lzip[T.1]	11.6
lzip[T.1]:sparse[T.1]	5.2
lzip[T.1]:seek[T.1]	7.5
lzip[T.1]:verbose[T.1]	7.5

Table A.1: The VIF Table Part 1 of system Tar after Data cleaning

features	VIF factor
lzip[T.1]:sortname[T.1]	7.5
level_9[T.1]:lzip[T.1]	11.6
level_5[T.1]:lzma[T.1]	16.5
lzma[T.1]:sparse[T.1]	5.4
lzma[T.1]:seek[T.1]	10.9
lzma[T.1]:verbose[T.1]	10.2
lzma[T.1]:sortname[T.1]	10.9
level_5[T.1]:lzop[T.1]	11.6
lzop[T.1]:sparse[T.1]	5.2
lzop[T.1]:seek[T.1]	7.5
lzop[T.1]:verbose[T.1]	7.5
lzop[T.1]:sortname[T.1]	7.5
level_9[T.1]:lzop[T.1]	11.6
gzip[T.1]:level_5[T.1]	11.6
gzip[T.1]:sparse[T.1]	5.2
gzip[T.1]:seek[T.1]	7.5
gzip[T.1]:verbose[T.1]	7.5
gzip[T.1]:sortname[T.1]	7.5
gzip[T.1]:level_9[T.1]	11.6
level_5[T.1]:zip[T.1]	10.9
seek[T.1]:zip[T.1]	7.5
verbose[T.1]:zip[T.1]	7.5
sortname[T.1]:zip[T.1]	7.5
level_9[T.1]:zip[T.1]	10.9
level_5[T.1]:zstd[T.1]	10.0
seek[T.1]:zstd[T.1]	8.9
verbose[T.1]:zstd[T.1]	8.3
sortname[T.1]:zstd[T.1]	8.9
level_5[T.1]:verify[T.1]	2.9
level_9[T.1]:verify[T.1]	2.9
oldgnu[T.1]:verify[T.1]	2.1
oldgnu[T.1]:sparse[T.1]	2.8
oldgnu[T.1]:seek[T.1]	3.6
oldgnu[T.1]:verbose[T.1]	3.6
oldgnu[T.1]:sortname[T.1]	3.6
pax[T.1]:verify[T.1]	2.1
pax[T.1]:sparse[T.1]	2.8
pax[T.1]:seek[T.1]	3.6
pax[T.1]:verbose[T.1]	3.6
pax[T.1]:sortname[T.1]	3.6
posix[T.1]:verify[T.1]	2.1
posix[T.1]:sparse[T.1]	2.8

Table A.2: The VIF Table Part 2 of system Tar after Data cleaning

features	VIF factor
posix[T.1]:seek[T.1]	3.6
posix[T.1]:verbose[T.1]	3.6
posix[T.1]:sortname[T.1]	3.6
ustar[T.1]:verify[T.1]	1.6
seek[T.1]:ustar[T.1]	3.1
ustar[T.1]:verbose[T.1]	2.8
sortname[T.1]:ustar[T.1]	3.1
v7[T.1]:verify[T.1]	1.6
seek[T.1]:v7[T.1]	3.1
v7[T.1]:verbose[T.1]	2.9
sortname[T.1]:v7[T.1]	3.1
seek[T.1]:sparse[T.1]	2.9
sortname[T.1]:sparse[T.1]	2.9
level_9[T.1]:two[T.1]:xz[T.1]	2.9
level_9[T.1]:lzma[T.1]:two[T.1]	2.9
level_9[T.1]:two[T.1]:zstd[T.1]	2.4
level_5[T.1]:sparse[T.1]:two[T.1]	2.4
level_5[T.1]:seek[T.1]:two[T.1]	4.9
level_5[T.1]:two[T.1]:verbose[T.1]	2.9
level_5[T.1]:sortname[T.1]:two[T.1]	4.9
level_9[T.1]:sparse[T.1]:two[T.1]	2.4
level_9[T.1]:seek[T.1]:two[T.1]	4.9
level_9[T.1]:two[T.1]:verbose[T.1]	2.9
level_9[T.1]:sortname[T.1]:two[T.1]	4.9
level_5[T.1]:sparse[T.1]:xz[T.1]	3.6
level_5[T.1]:seek[T.1]:xz[T.1]	8.1
level_5[T.1]:verbose[T.1]:xz[T.1]	7.8
level_5[T.1]:sortname[T.1]:xz[T.1]	8.1
level_9[T.1]:sparse[T.1]:xz[T.1]	3.6
level_9[T.1]:seek[T.1]:xz[T.1]	8.1
level_9[T.1]:verbose[T.1]:xz[T.1]	7.8
level_9[T.1]:sortname[T.1]:xz[T.1]	8.1
level_5[T.1]:lzip[T.1]:sparse[T.1]	4.0
level_5[T.1]:lzip[T.1]:seek[T.1]	5.2
level_5[T.1]:lzip[T.1]:verbose[T.1]	5.2
level_5[T.1]:lzip[T.1]:sortname[T.1]	5.2
level_9[T.1]:lzip[T.1]:sparse[T.1]	4.0
level_9[T.1]:lzip[T.1]:seek[T.1]	5.2
level_9[T.1]:lzip[T.1]:verbose[T.1]	5.2
level_9[T.1]:lzip[T.1]:sortname[T.1]	5.2
level_5[T.1]:lzma[T.1]:sparse[T.1]	5.4
level_5[T.1]:lzma[T.1]:seek[T.1]	8.1

Table A.3: The VIF Table Part 3 of system Tar after Data cleaning

features	VIF factor
level_5[T.1]:lzma[T.1]:verbose[T.1]	7.8
level_5[T.1]:lzma[T.1]:sortname[T.1]	8.1
level_9[T.1]:lzma[T.1]:sparse[T.1]	5.4
level_9[T.1]:lzma[T.1]:seek[T.1]	8.1
level_9[T.1]:lzma[T.1]:verbose[T.1]	7.8
level_9[T.1]:lzma[T.1]:sortname[T.1]	8.1
level_5[T.1]:lzop[T.1]:sparse[T.1]	4.0
level_5[T.1]:lzop[T.1]:seek[T.1]	5.2
level_5[T.1]:lzop[T.1]:verbose[T.1]	5.2
level_5[T.1]:lzop[T.1]:sortname[T.1]	5.2
level_9[T.1]:lzop[T.1]:sparse[T.1]	4.0
level_9[T.1]:lzop[T.1]:seek[T.1]	5.2
level_9[T.1]:lzop[T.1]:verbose[T.1]	5.2
level_9[T.1]:lzop[T.1]:sortname[T.1]	5.2
gzip[T.1]:level_5[T.1]:sparse[T.1]	4.0
gzip[T.1]:level_5[T.1]:seek[T.1]	5.2
gzip[T.1]:level_5[T.1]:verbose[T.1]	5.2
gzip[T.1]:level_5[T.1]:sortname[T.1]	5.2
gzip[T.1]:level_9[T.1]:sparse[T.1]	4.0
gzip[T.1]:level_9[T.1]:seek[T.1]	5.2
gzip[T.1]:level_9[T.1]:verbose[T.1]	5.2
gzip[T.1]:level_9[T.1]:sortname[T.1]	5.2
level_5[T.1]:sparse[T.1]:zip[T.1]	2.2
level_5[T.1]:seek[T.1]:zip[T.1]	5.2
level_5[T.1]:verbose[T.1]:zip[T.1]	5.2
level_5[T.1]:sortname[T.1]:zip[T.1]	5.2
level_9[T.1]:sparse[T.1]:zip[T.1]	2.2
level_9[T.1]:seek[T.1]:zip[T.1]	5.2
level_9[T.1]:verbose[T.1]:zip[T.1]	5.2
level_9[T.1]:sortname[T.1]:zip[T.1]	5.2
level_5[T.1]:sparse[T.1]:zstd[T.1]	3.6
level_5[T.1]:seek[T.1]:zstd[T.1]	7.4
level_5[T.1]:verbose[T.1]:zstd[T.1]	7.2
level_5[T.1]:sortname[T.1]:zstd[T.1]	7.4
level_9[T.1]:sparse[T.1]:zstd[T.1]	3.6
level_9[T.1]:seek[T.1]:zstd[T.1]	7.4
level_9[T.1]:verbose[T.1]:zstd[T.1]	7.2
level_9[T.1]:sortname[T.1]:zstd[T.1]	7.4
workload	1.0

Table A.4: The VIF Table Part 4 of system Tar after Data cleaning

features	VIF factor
Intercept	32.8
trendy[T.1]	3.1
frumpy[T.1]	3.1
crafty[T.1]	3.1
jumpy[T.1]	3.1
handy[T.1]	3.1
transExtWeight[T.1]	1.5
transExtCard[T.1]	1.5
transExtScc[T.1]	1.5
heuristicVmtf[T.1]	12.0
heuristicVsids[T.1]	12.0
enumModeRecord[T.1]	12.0
enumModeCautious[T.1]	12.0
parallel_mode_2[T.1]	11.9
parallel_mode_4[T.1]	11.9
enumModeRecord[T.1]:heuristicVmtf[T.1]	10.6
enumModeRecord[T.1]:parallel_mode_2[T.1]	10.6
heuristicVmtf[T.1]:parallel_mode_2[T.1]	10.6
enumModeRecord[T.1]:parallel_mode_4[T.1]	10.6
heuristicVmtf[T.1]:parallel_mode_4[T.1]	10.6
enumModeRecord[T.1]:heuristicVsids[T.1]	10.6
heuristicVsids[T.1]:parallel_mode_2[T.1]	10.6
heuristicVsids[T.1]:parallel_mode_4[T.1]	10.6
enumModeCautious[T.1]:heuristicVmtf[T.1]	10.6
enumModeCautious[T.1]:parallel_mode_2[T.1]	10.6
enumModeCautious[T.1]:parallel_mode_4[T.1]	10.6
enumModeCautious[T.1]:heuristicVsids[T.1]	10.6
parallel_mode_2[T.1]:trendy[T.1]	2.0
parallel_mode_4[T.1]:trendy[T.1]	2.0
frumpy[T.1]:parallel_mode_2[T.1]	2.0
frumpy[T.1]:parallel_mode_4[T.1]	2.0
crafty[T.1]:parallel_mode_2[T.1]	2.0
crafty[T.1]:parallel_mode_4[T.1]	2.0
jumpy[T.1]:parallel_mode_2[T.1]	2.0
jumpy[T.1]:parallel_mode_4[T.1]	2.0
handy[T.1]:parallel_mode_2[T.1]	2.0
handy[T.1]:parallel_mode_4[T.1]	2.0
enumModeRecord[T.1]:heuristicVmtf[T.1]:parallel_mode_2[T.1]	7.7
enumModeRecord[T.1]:heuristicVmtf[T.1]:parallel_mode_4[T.1]	7.7
enumModeRecord[T.1]:heuristicVsids[T.1]:parallel_mode_2[T.1]	7.7
enumModeRecord[T.1]:heuristicVsids[T.1]:parallel_mode_4[T.1]	7.7
enumModeCautious[T.1]:heuristicVmtf[T.1]:parallel_mode_2[T.1]	7.7
enumModeCautious[T.1]:heuristicVmtf[T.1]:parallel_mode_4[T.1]	7.7
enumModeCautious[T.1]:heuristicVsids[T.1]:parallel_mode_2[T.1]	7.7
enumModeCautious[T.1]:heuristicVsids[T.1]:parallel_mode_4[T.1]	7.7
workload	1.0

Table A.5: The VIF Table of system Clasp after Data cleaning

BIBLIOGRAPHY

[1] Aylin Alin. "Multicollinearity." In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (2010), pp. 370–374.

[2] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. "Sampling Effect on Performance Prediction of Configurable Systems: A Case Study." In: *Proceedings of the International Conference on Performance Engineering (ACM/SPEC)*. Association for Computing Machinery, 2020, pp. 277–288.

[3] Tomáš Balyo, Nils Froleyks, Marijn J.H. Heule, Markus Iser, Matti Järvisalo, and Martin Suda, eds. *Proceedings of SAT Competition 2020: Solver and Benchmark Descriptions*. Department of Computer Science Report Series B. Department of Computer Science, University of Helsinki, 2020.

[4] Jamal I. Daoud. "Multicollinearity and Regression Analysis." In: 949 (2017), p. 012009.

[5] Johannes Dorn, Sven Apel, and Norbert Siegmund. "Mastering Uncertainty in Performance Estimations of Configurable Software Systems." In: *Proceedings of the International Conference on Automated Software Engineering (IEEE/ACM)*. Association for Computing Machinery, 2020, pp. 684–696.

[6] Ammar Abbas Elmas. "Investigation of Single-Rate Triangular 3d Mesh Compression Algorithms." Master of Science. Çukurova University, 2019, p. 115.

[7] Pooyan Jamshidi, Norbert Siegmund, Miguel Velez, Christian Kästner, Akshay Patel, and Yuvraj Agarwal. "Transfer Learning for Performance Modeling of Configurable Systems: An Exploratory Analysis." In: *Proceedings of the International Conference on Automated Software Engineering (IEEE/ACM)*. IEEE Computer Society, 2017, pp. 497–508.

[8] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. "Distance-Based Sampling of Software Configuration Spaces." In: *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE / ACM, 2019, pp. 1084–1094.

[9] Oddur Kjartansson, Alexander Gutkin, Alena Butryna, Isin Demirsahin, and Clara Rivera. "Open-Source High Quality Speech Datasets for Basque, Catalan and Galician." In: *Proceedings of the Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*. European Language Resources association (ELRA), 2020, pp. 21–27.

[10] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. "HMDB: A Large Video Database for Human Motion Recognition." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011.

[11] Mirella Lapata. "Automatic Evaluation of Information Ordering: Kendall's Tau." In: *Computational Linguistics* 32 (2006), pp. 471–484.

[12] Lizhi Liao, Jinfu Chen, Heng Li, Yi Zeng, Weiyi Shang, Jianmei Guo, Catalin Sporea, Andrei Toma, and Sarah Sajedi. "Using Black-Box Performance Models to Detect Performance Regressions under Varying Workloads: An Empirical Study." In: *Empirical Software Engineering* 25 (2020), pp. 4130–4160.

[13] Filippo Mantovani, Marta Garcia-Gasulla, José Gracia, Esteban Stafford, Fabio Banchelli, Marc Josep-Fabrego, Joel Criado-Ledesma, and Mathias Nachtmann. "Peformance and Energy Consumption of HPC Workloads on a Cluster based on Arm ThunderX2 CPU." In: *Future Generation Computer Systems* 112 (2020).

[14] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation." In: *Computer Vision and Pattern Recognition*. 2016.

[15] Daniele Rogora, Antonio Carzaniga, Amer Diwan, Matthias Hauswirth, and Robert Soulé. "Analyzing System Performance with Probabilistic Performance Annotations." In: *Proceedings of the European Conference on Computer Systems (EuroSys)*. Association for Computing Machinery, 2020, 43:1–43:14.

[16] Samuel Kounev and Klaus-Dieter Lange and Jóakim von Kistowski. *Systems Benchmarking: For Scientists and Engineers*. Springer, 2020, pp. 1–22, 185–202.

[17] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. "Performance-Influence Models for Highly Configurable Systems." In: *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. Association for Computing Machinery, 2015, pp. 284–294.

[18] Qiang Xu, Y. Charlie Hu, and Abhilash Jindal. "App Parameter Energy Profiling: Optimizing App Energy Drain by Finding Tunable App Parameters." In: *Computing Research Repository (CoRR)* abs/2009.12156 (2020).