University of Passau

Department of Informatics and Mathematics

Master's Thesis

Comparison of Analytical and Empirical Performance Models: A Case Study on Multigrid Systems

Author:

Christian Kaltenecker

November 29, 2016

Advisors:

Prof. Dr.-Ing. Sven Apel

Chair of Software Engineering

Alexander Grebhahn

Chair of Software Engineering

Kaltenecker, Christian:
Comparison of Analytical and Empirical Performance Models: A Case Study on Multigrid Systems
Master's Thesis, University of Passau, 2016.

Abstract

Runtime complexity of software can be described by the $\mathcal{O}$-notation. However, this is only a theoretical indicator and cannot be used as an indicator for the actual runtime of a software. For the approximation of the runtime of a certain software, so-called performance models are used, which can be categorized in analytical and empirical performance models. While analytical performance models are created using domain knowledge, empirical performance models are obtained using machine-learning strategies. In this work, we compare two analytical and empirical performance models for two multigrid systems, SMG2000 and BoomerAMG. The empirical performance models are generated by the tool SPL Conqueror using different sampling heuristics to define the learning set. To allow the comparison between these kinds of performance models, we propose two different comparison strategies, which we evaluate and discuss in this thesis. For one comparison strategy, we investigate on different distance and similarity measures. We observe that the selection of the sampling heuristic has an influence on both comparison strategies. Furthermore, we notice that both comparison strategies as well as the results of the distance and similarity measures differ from each other.

Contents

Contents vii

List of Figures ix

List of Tables xi

List of Algorithms xiii

1 Introduction 1

2 Background 3
2.1 Performance Models . . . 3
2.1.1 Analytical Performance Models . . . 4
2.1.2 Empirical Performance Models . . . 5
2.2 SPL Conqueror . . . 5
2.2.1 Performance-Influence Models . . . 6
2.2.2 Sampling Heuristics . . . 8
2.3 Similarity and Distance Measures . . . 12
2.3.1 Cosine Similarity . . . 13
2.3.2 Jaccard . . . 13
2.3.3 Minkowski . . . 13
2.4 Multigrid . . . 14
2.4.1 Overview . . . 14
2.4.2 Grid . . . 14
2.4.3 Restriction and Prolongation . . . 15
2.4.4 Smoothing . . . 16
2.4.5 Solver . . . 17

3 Case Studies 19
3.1 SMG2000 . . . 19
3.1.1 Functionality . . . 19
3.1.2 Analytical Performance Models . . . 20
3.2 BoomerAMG . . . 22
3.2.1 Functionality . . . 22
3.2.2 Analytical Performance Model . . . 22

4 Analyzation 25
4.1 Syntactical Comparison . . . 25

4.2 Semantical Comparison 26

5 Evaluation **29**
5.1 Setup 29
5.1.1 Cluster 29
5.1.2 Software 30
5.2 Obtaining Parameters for the Evaluation 31
5.2.1 SMG2000 31
5.2.2 BoomerAMG 32
5.2.3 Benchmarks 32
5.3 Learning Performance-Influence Models 33
5.4 Comparison 34
5.4.1 SMG2000 34
5.4.2 BoomerAMG 43
5.5 Discussion 46
5.6 Threats to Validity 49

6 Related Work **51**

7 Conclusion **53**
7.1 Summary 53
7.2 Future Work 54

A Appendix **57**
A.1 Folder and File Structure on the CD 57

Bibliography **59**

List of Figures

2.1 General sketch of SPL Conqueror's approach 6
2.2 Selection of numeric values using the Full Factorial design 9
2.3 Generation of a Central Composite Design 11
2.4 Selection of numeric values using the Central Composite Design . . . 11
2.5 V-cycle in detail 15
2.6 Restriction in detail 15
2.7 Prolongation from a single grid cell to a 2×2 grid. 16
2.8 Error in a grid when using a solver 17

3.1 Allocation of grids to processes 20
3.2 Example of an unstructured grid 22

4.1 Semantic comparison of the exemplary performance models 27

5.1 Results of the 1-dimensional grid of SMG2000 35
5.2 Improved results of the 1-dimensional grid of SMG2000 36
5.3 Results of the 2-dimensional grid of SMG2000 38
5.4 Improved results of the 2-dimensional grid of SMG2000 38
5.5 Results of the performance models and the measurements on 3-dimensional grids of SMG2000 41
5.6 Improved results of the analytical performance model on 3-dimensional grids of SMG2000 42
5.7 Results of the baseline model in relation of BoomerAMG. 45

A.1 Folder hierarchy on the CD 57
A.2 Files in the BoomerAMG folder 58
A.3 Files in the folders of SMG2000 58

List of Tables

2.1 Candidate sets for three numeric options 8

2.2 Plackett-Burman with parameters $(n = 9, l = 3)$ 10

4.1 Exemplary results of the distance and similarity measures 28

5.1 Processor topology ranges for SMG2000 31

5.2 Grid size range for SMG2000 on Chimaira 32

5.3 Numeric ranges for the grid size of BoomerAMG 32

5.4 Benchmark results for different number of nodes on Chimaira 33

5.5 Configurations of the Plackett-Burman design 33

5.6 Comparison results of different sampling heuristics in SMG2000 with an 1-dimensional grid 35

5.7 Results of the comparison of the 1-dimensional analytical performance model with the measurements 36

5.8 Comparison results of different sampling heuristics in SMG2000 with a 2-dimensional grid 38

5.9 Results of the comparison of the analytical performance model with the measurements from the 2-dimensional case 39

5.10 Best and worst results of the syntactical and semantical comparison strategy in the 3-dimensional grid. 41

5.11 Results of the comparison of the analytical performance model with the measurements from the 3-dimensional case 42

5.12 Results of the syntactical and semantical comparison strategies in BoomerAMG 45

5.13 Results of the comparison of the analytical performance model with the measurements from BoomerAMG 46

List of Algorithms

1 Sketch of the Feature Forward Selection of SPL Conqueror 6

1. Introduction

One simple way to express runtime complexity of a software is to use the $\mathcal{O}$-notation. However, this notation provides only information about the theoretical complexity and not about the real runtime of a software. Information about the runtime of a software is important when the software is executed on shared computers. For instance, computer clusters and supercomputers are used by multiple persons. To this end, each person has a certain time limit for the execution of its tasks. To tailor software to specific needs, most software provides configuration options, which allow the selection or deselection of components. These configuration options can have an impact on the performance, which give the estimation of the runtime more importance. The estimation of the runtime can be used to find a performance-optimal configuration to maximize efficiency of the executed software. A famous algorithm in the domain of high performance computing is multigrid, which is used to solve discretized PDEs. In multigrid systems, components such as the solver, smoother and number of pre-smoothing and post-smoothing steps can be configured. Each component has a different effect on the performance of the system as described by Trottenberg et al. [TS01].

However, finding the performance-optimal configuration may turn out to be an infeasible challenge if every software configuration is measured in a brute-force manner. The disadvantage in measuring in a brute-force manner is the exponentially high number of software configurations with respect to the configuration parameters. To overcome this problem, performance models can be used to predict the runtime of a certain software configuration on the targeted hardware. For the prediction of the execution time, the influence factors of different system aspects with the highest impact on the performance of the software are considered. Considerable aspects on the computing system are characteristics of hardware and software. For instance, the different cache levels and the clock rate of the central processing unit (CPU) are relevant hardware aspects, whereas configuration options are software characteristics. Moreover, these aspects can interact with each other, which means that these aspects together imply a certain influence on the overall performance.
According to their nature, performance models can be categorized in two groups,

namely *analytical performance models* and *empirical performance models*. The former are created by domain experts that use their own perception of the program to create such analytical performance models. The latter can be created by using a machine-learning algorithm that is based on a small set of measurements, so-called empirical performance models.

In this work, we focus on the comparison of analytical performance models and empirical performance models of two multigrid programs, SMG2000 [Car01] and BoomerAMG [Yan02]. The empirical performance models we use in this thesis are generated by the tool SPL Conqueror by using different sampling strategies. Throughout this study, we address multiple issues that arise regarding the creation of empirical performance models. The research questions we examine in this thesis, are the following:
RQ1: Is it possible to identify the influences proposed by analytical performance models?
RQ2: Does the sampling strategy have a high influence on the similarity of empirical performance-influence models to analytical performance models?
RQ3: Is it beneficial to compare analytical and empirical performance models using only syntactic information?
RQ4: Is it beneficial to compare analytical and empirical performance models using distance and similarity measures?
RQ5: Do the results of the similarity and distance measures correlate with the error rate?

This thesis is structured as follows:

Firstly, in Chapter 2 we provide general information, which is meant to improve the understanding of the used software and terminology. We provide information about the properties of analytical performance models and empirical performance models, SPL Conqueror, the software that is used to generate empirical performance models, and introduce the theory behind multigrid systems as well as the distance and similarity measures.

In Chapter 3, the multigrid case studies SMG2000 and BoomerAMG are presented in more detail. For both multigrid case studies, we present characteristics, the functionality, and we describe their analytical performance models in this chapter.

Based on performance models, we propose different comparison strategies to compare multiple different performance models in Chapter 4.

The evaluation of the performance models and the results of the different types of comparisons are presented in Chapter 5. First, we show the setup of the measurements. Thereafter, we use these performance models to analyze and compare them with each other on the base of two case studies, SMG2000 and BoomerAMG. The outcomes are used to answer the research questions.

In Chapter 6, we present other publications that are concerned with the generation or the improvement of analytical and empirical performance models.

Last, in Chapter 7, we conclude this thesis and present future work.

2. Background

In this chapter, we provide general information on the terminology and software of this study. We begin with explaining the performance models in Section 2.1, where we give a deeper insight into what performance models are, what they are used for, and how they can be categorized. Afterwards, we roughly describe the basic concepts of SPL Conqueror, which is used to generate the empirical performance models in this thesis, in Section 2.2. Then, the mathematical properties of distance and similarity measures are shown in Section 2.3, which are used in the semantical comparison to express the similarity or dissimilarity of two performance models. Finally, in Section 2.4, we provide basic knowledge of multigrids to understand the behavior of the case studies as well as the predicted influences of the performance models.

2.1 Performance Models

Our study focuses on the performance prediction of software, due to the fact that a high number of configuration options would also mean a high effort for measuring every configuration of a software. The configurations of the software are measured to gain knowledge about the performance behavior of this software and to find the performance-optimal configuration. To minimize the effort of predicting the runtime of a software, mathematical equations are used, whose variables are the different hardware factors, software factors and interactions among them. These equations can be used to predict the performance of a system and thus help to find the performance-optimal configuration of a software. Mathematical equations that are used to predict the performance of a given configuration of a software are called *performance models*. The result of such models is usually expressed in seconds or milliseconds. Since both case studies contain hardware characteristics, such as delay when interprocessor messages are exchanged, not the CPU time but the wall-clock time is considered by the performance models. Moreover, in the CPU time hardware delays are not considered. According to their method by which they are created, such models can be categorized as follows:

- Analytical Performance Models
- Empirical Performance Models

The difference between these kind of models stems from the information they are based on. The creation of analytical performance models is a white-box approach by including the domain knowledge of experts, whereas the empirical performance models are built in a black-box manner by including only a set of performance tests, where different configurations of the software are measured [DR15]. In Didona et al. [DR15] so-called gray-box modeling techniques are proposed, which are hybrid approaches that are based on the information of empirical performance models as well as analytical performance models. However, we only focus on analytical performance models and empirical performance models in this work, which we describe in the following.

2.1.1 Analytical Performance Models

Developers often want to check if their perceptions of the written software are correct and write performance models to look out for abnormal deviations in the performance, which would mean that the software behaves unexpectedly. Such models are created by one or more domain experts and are called analytical performance models. In the last decades, research was focused on the creation of this kind of performance models, see for example the work of Tay et al. [Tay13]. In general, this is a white-box approach, as the domain expert includes knowledge about the software and their perception of it [DR15].
Furthermore, this kind of model can be used to predict the performance of a software on different architectures. To this end, they include information of the underlying hardware, such as the floating point operations per second (Flops) to consider the architecture in the performance prediction [Tay13].
Besides, such analytical performance models can become very complex, the more influence factors they consider. Hence, considering a high amount of factors makes it more difficult for the domain expert to create such performance models. Moreover, a more complex equation makes it difficult for the user to understand and to use the analytical performance models. To overcome the problem of complex analytical performance models, only a part of the software and of the environment it is executed in is modeled. Possible factors that have an impact on the performance are for example the cache-miss rate and the temperature of the CPU [MDHS09]. Taking these factors in consideration, would make the analytical performance model far too complex. To overcome this problem, the factors with the highest impact on the performance are considered by the analytical performance model, which leads to a minor error in the performance prediction.
Nonetheless, analytical performance models are needed in every domain where the prediction of the performance is crucial and not exclusively in the domain of high-performance computing. For instance, in the domain of distributed computing an analytical performance model for MapReduce was created by Yang et al. [YS11]. Another example is the analytical performance model made by Hong et al. [HK09] for graphics processing unit (GPU) architectures.

To sum it up, analytical performance models are portable and can be applied to different architectures, but include a certain error, as they are only considering the most relevant influence factors to the performance. Furthermore, if the perceptions of the domain experts are wrong, the analytical performance models produce wrong predictions and hence are not helpful.

2.1.2 Empirical Performance Models

Machine-learning algorithms can be used to predict the performance of a given software configuration [Bis06]. If one is interested only in the optimal configuration of the software and not the impact of the configuration options on the performance, then different machine-learning approaches can be used, such as genetic algorithms [MSB91] and gradient descent [Bot10]. More powerful tools like CatWalk [WBH+14] and SPL Conqueror [SRK+12] are capable of estimating the influence of the most relevant configuration options and the most relevant configuration-option interactions on the performance of the system. This estimation is done by using machine-learning algorithms on a set of measurements of the target system. The empirical performance models in this thesis are created in a black-box manner, as no knowledge about the system itself is included in the models [DR15]. Unlike analytical performance models, this kind of performance model only considers the hardware on which the measurements were executed on. To this end, mostly a higher prediction accuracy is achieved with respect to analytical performance models, but these predictions are not portable. As analytical performance models are created by domain experts, empirical performance models can be created based on measurements of the respective software and a tool, such as CatWalk and SPL Conqueror and do not expect background knowledge of the software.

2.2 SPL Conqueror

In this section, we present the concepts of SPL Conqueror, which is used to learn empirical performance models in this thesis. Besides, in the publications related to SPL Conqueror, the empirical performance models are called empirical performance-influence models. Since we use SPL Conqueror, we adapt this terminology and call it from now on empirical performance-influence model. The goal of this program is to quantify influences of options and their interactions to ease understanding, debugging and optimization of the configurable systems [SRK+12]. The sketch of SPL Conqueror's approach is shown in Figure 2.1. To minimize the number of measurements for the quantification of the influences, different sampling heuristics can be used in SPL Conqueror, where some are presented later and used in the evaluation. To learn an empirical performance-influence model, SPL Conqueror needs a set of measurements to consider, which we call *Learning Set* in the following. Based on this learning set, SPL Conqueror uses *Feature Forward Selection* [CS14], which is an iterative machine-learning algorithm. The influence of different models, each of which considers the influence of one single configuration, is computed by using *Stepwise Linear Regression*. Then, the best model with respect to the error rate is selected and another set of models, called candidates in the following, is generated on the base of the best model. The different generated models now contain interactions between configuration options. The next step is again the computation

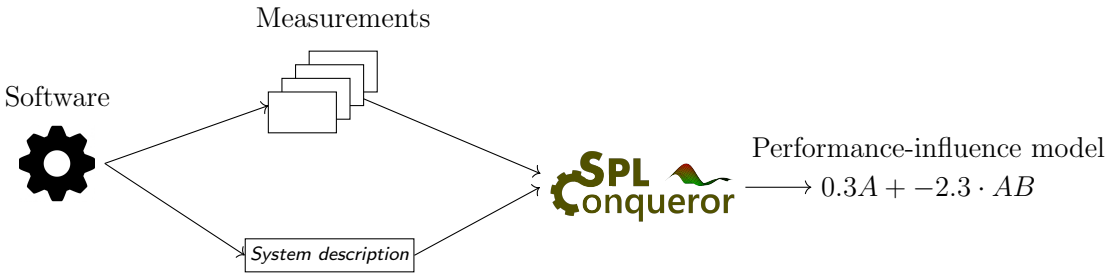

Figure 2.1: General sketch of SPL Conqueror's approach, which needs the measurements as well as the corresponding system description of a software to produce an empirical performance-influence model.

Algorithm 1: Sketch of the Feature Forward Selection of SPL Conqueror. Taken from [SGAK15].

```
Data: measurements, 𝒪
Result: performance model
optionSet = ∅, error = ∞
repeat
    lastError = error
    bestCandidate = ⊥
    candidates = generateCandidates(optionSet, 𝒪);
    foreach option in candidates do
        model = learnFunction(optionSet ∪ {option},
                              measurements)
        modelError = computeError(model, measurements)
        if modelError < error then
            error = modelError, bestCandidate = candidate
        end
    end
    if bestCandidate ≠ ⊥ then
        optionSet = optionSet ∪ {option}
    end
until (lastError − error < margin) ∨ (error < threshold);
return learnFunction(optionSet, measurements);
```

of the influence. These steps are repeated iteratively until the error rate has reached a certain value or the improvement from one round to another is under a certain threshold.
We address one configuration option or an interaction of configuration options with its influence as *Term* throughout this work.

In Section 2.2.1, we present the empirical performance-influence models that are generated by SPL Conqueror. For the generation of different learning sets, different sampling heuristics are used, which we present in Section 2.2.2. We use these sampling heuristics in this thesis and compare them with each other in relation to the analytical performance models.

2.2.1 Performance-Influence Models

In this section, we describe performance-influence models according to the work of Siegmund et al. [SGAK15]. Assume that $\mathcal{O}$ is the set of all configuration options and $\mathcal{C}$ the set of all valid configurations. Furthermore, a configuration $c \in \mathcal{C}$ is a function $c : \mathcal{O} \to \mathbb{R}$ that assigns a value to every configuration option. For a binary configuration option $b \in \mathcal{O}$, $c(b) = 1$ if the respective option is selected and $c(b) = 0$

if not. In contrast to the binary options, the range of a numeric options consists of more than one value, although only one value can be selected for one single configuration. It is assumed that every value of the numeric option has a different influence upon the final performance and that this influence can be represented as a function. Additionally, configuration options can interact with each other and have a high influence on the performance. For instance, the compression can have a positive effect on encryption, as the encryption takes less time on a smaller set of information.
A performance-influence model is a function $\prod : \mathcal{C} \rightarrow \mathbb{R}$ that can be written as a sum of terms over configuration values. In one term of the model, different functions can be composed of the configuration options of each term. This composition results in shapes such as $\beta \cdot c(o)$, $\beta \cdot c(o)^2$, $\beta \cdot \sqrt{c(o)}$, where β is the influence of the configuration option upon the performance and $c(o)$ is the value of option o in the configuration c. The influence β is computed by using linear regression. Besides, SPL Conqueror also finds terms with multiple configuration options like $\beta \cdot c(o_1) \cdot c(o_2)$ with the influence of the interaction symbolized as β and configuration options o_1, $o_2 \in \mathcal{C}$. Furthermore, a *term* is an expression $\beta \cdot c(o_1)$ or $\beta \cdot c(o_1) \cdot c(o_2) \cdots c(o_n)$ where o_1, o_2, $\ldots o_n \in \mathcal{O}$. Moreover, a term that refers to a single configuration option o is denoted as ϕ_o, whereas interactions of multiple options $i..j$ is denoted as $\phi_{i..j}$.

To sum it up, every performance-influence model is of the form:

$$\prod(c) = \beta_0 + \sum_{i \in \mathcal{O}} \phi_i(c(i)) + \sum_{i..j \in \mathcal{O}} \phi_{i..j}(c(i)..c(j))$$

In the equation, β_0 is a value that is related to no specific configuration option or in other words is present in every software configuration and $\sum_{i \in \mathcal{O}} \phi_i(c(i))$ stands for the sum of all individual configuration options, whereas $\sum_{i..j \in \mathcal{O}} \phi_{i..j}(c(i)..c(j))$ denotes the sum of all configuration option interactions.

To find such a model, multiple steps are performed by using *Stepwise Linear Regression*, as shown in Algorithm 1. In the following, we describe this algorithm in more detail.

In Lines $2-17$ of Algorithm 1, the best model is searched. This is done by *Feature Forward Selection* by adding each found model to the current model for improving the current model. To choose among different models, a set of candidates including the single configuration options and interactions of them have to be generated in Line 5. These candidates differ from each other by the type of influence of the considered configuration options (e.g., logarithmic influence, linear influence). Then, in Lines $6-13$, the candidates are added iteratively to the model. Thereby, the influence of each term is computed and afterwards the error rate of the model to the measurement results. If an improvement was made, then the model is selected as the best candidate. Note, that if none of the candidates is better than the best model from the last round, then the best model will be selected in Lines $14-16$. In this step, the influences of every term of the model are recomputed. This procedure is repeated until the error has reached a certain value (*threshold*) and the improvement from one model to another model is greater than a value (*margin*). Finally, the best model is returned in Line 19 as the result of the algorithm.

2.2.2 Sampling Heuristics

Until now, we presented the algorithm of SPL Conqueror to learn performance-influence models that are based on a learning set. Due to the fact that measuring every valid configuration would take a lot of time, only a set of configurations should be used for creating a suitable empirical performance-influence model. To determine a representative learning set of the set of all software configurations, sampling heuristics are used. Since we have two types of configuration options, namely binary and numeric, different sampling heuristics are used for the sampling of binary options and others for the sampling of numeric options. Nevertheless, only numeric-option sampling is considered, as only numeric configuration options are relevant in the analytical performance model of our multigrid case studies, SMG2000 and BoomerAMG. To express the value range of a numeric option, we use the notation `[<minValue>..<maxValue>;<stepSize>]`, where `minValue` is the lower bound, `maxValue` represents the upper bound and `stepSize` the step size of the numeric option.

In this work, we use the following sampling heuristics to generate the empirical performance-influence models:

- ***Full Factorial Design***
- ***Random Design***
- ***Plackett-Burman Design***
- ***Central Composite Design***

Table 2.1: Matrix of candidates for three numeric options $A = [1..2; 1]$, $B = [1..5; 4]$ and $C = [1..2; 1]$. The first matrix of candidates, ξ_8 contains every configuration, whereas ξ_2 contains only 2 of these configurations, which are selected by the sampling heuristic.

$\xi_8 =$

A	B	C
1	1	1
1	1	2
1	5	1
1	5	2
2	1	1
2	1	2
2	5	1
2	5	2

$\xi_2 =$

A	B	C
1	1	1
2	5	2

Each of these heuristics are *Experimental Designs* [Puk06]. An experimental design is an approach to find a subset from the set of all configurations. Instead of measuring every configuration of a software, experimental designs generate a candidate matrix that consists of software configurations which have to be measured to make conclusions. According to de Aguiar et al. [dABK+95], such a candidate matrix contains the software configurations that have to be measured. The candidate matrix is denoted as ξ_i, where i is the number of chosen configurations of the sampling

heuristic. An example for the candidate matrix is shown in Table 2.1. Every column of the matrix of candidates stands for a configuration option and every line describes a configuration. The matrix of candidates containing all configurations is usually depicted as ξ_N where N is the total number of configurations. Furthermore, from this matrix of candidates, further subsets ξ_n can be derived, where $0 < n \leq N$. Besides, some of the experimental designs such as the Plackett-Burman design create a matrix of candidates according to predefined shapes.

In the following, we describe the sampling heuristics that are used in this work more precisely.

Full Factorial Design. The Full Factorial design is the most simple experimental design, as it uses the whole configuration set as learning set and not a subset of it [BTA10]. This means that each valid configuration of the software is needed for the use of this design. The advantage of this design is that influences that appear only in a few configurations are considered. For calculating the number of needed measurements for the learning set of this design, the product of the cardinality of each configuration option is built. In Figure 2.2, we show the configurations that are measured by using this design for two configuration options A and B. Because of the high number of needed measurements, this design is less frequently used.

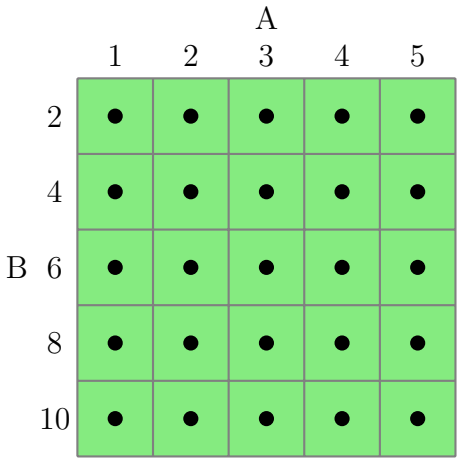

Figure 2.2: Selection of numeric values with the Full Factorial design using two numeric options A and B where $A = [1..5; 1]$ and $B = [2..10; 2]$.

Random Design. The Random design chooses random values from the value domains of the numeric options. This is possible, as we later define the lower and the upper bound of each numeric option as well as the step size between valid numeric option values. The used implementation of this design chooses the set of n configurations in a random manner by selecting a value of the range of each numeric option starting with an initial random seed. Since this sampling is random, the effectiveness of choosing good sampling may vary according to the chosen seed for the random-number generation.

Plackett-Burman Design. This design was proposed in 1946 by Plackett and Burman [PB46]. It is an experimental design that aims to minimize the variance of the estimates of variables that are independent and do not interact with each other.

Table 2.2: The Plackett-Burman Design for 8 numeric options (A - H) and the parameters ($n = 9, l = 3$). We use a mapping, where 0 is the minimum value, 2 the maximum value and 1 the midpoint of the numeric value range. Each row in the table depicts one configuration. From this software, 9 different configurations would be tested.

	A	B	C	D	E	F	G	H
1	0	1	1	2	0	2	2	1
2	1	0	1	1	2	0	2	2
3	2	1	0	1	1	2	0	2
4	2	2	1	0	1	1	2	0
5	0	2	2	1	0	1	1	2
6	2	0	2	2	1	0	1	1
7	1	2	0	2	2	1	0	1
8	1	1	2	0	2	2	1	0
9	0	0	0	0	0	0	0	0

In contrast to other designs, the Plackett-Burman design uses only a very limited number of measurements as learning set, which is also pre-defined. Furthermore, one can set the number of *levels* (3, 5 and 7), which affects the chosen number of distinct values from each numeric option. Based on the number of *levels* (l), the number of measurements n can be chosen (for $l = 3$, $n = \{9; 27; 81\}$, for $l = 5$, $n = \{25; 125\}$ and for $l = 7$, $n = \{49\}$). Depending on the chosen value of n and l, a specific predefined initial seed is selected to define the first configuration used by the design. Every further configuration except the last configuration is a shift to the right of the previous configuration. An additional configuration is added, where all minimum values of the numeric options are used.

For instance, in Table 2.2 we show the Plackett-Burman for ($n = 9, l = 3$) defining 9 measurements for 8 independent variables (numeric options A-H) with 3 levels. The values 0, 1 and 2 are mapped to the minimum, the middle and the maximum value of the configuration option, respectively. The pre-defined seed is 01120221 for the configuration ($n = 9, l = 3$).

Central Composite Design. Box and Wilson proposed the formerly known Box-Wilson Central Composite design in 1951 [BW51]. This experimental design expands the 2^k factorial design, which takes the minimum and the maximum values of each numeric option. The configurations chosen by the factorial design are called factorial points. In the factorial design, 2^k combinations are generated when having k numeric options. The Central Composite Design(CCD) adds $2 \cdot k$ further configurations, also called star points, as well as the midpoint of the configuration space to the learning set. This expansion is also shown in Figure 2.3 for the case $k = 2$. The star points have a certain distance to the center, which is depicted as α.

The form presented in Figure 2.3 is called *Central Composite Circumscribed*(CCC). In this design type, the distance from the center to the star points is depicted as α. Depending on the value of α, additional types of the *Central Composite Design*, namely *Central Composite Inscribed*(CCI) and *Central Composite Face Cen-*

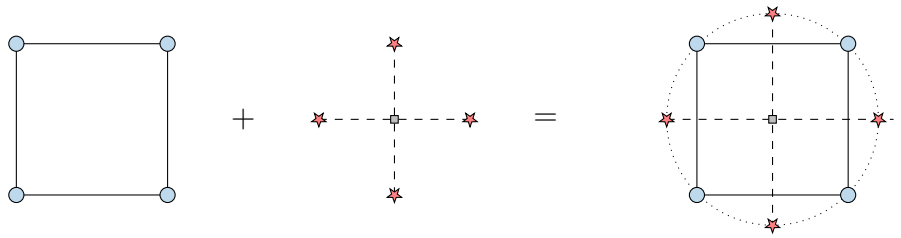

Figure 2.3: Generation of a Central Composite Design(CCD) for $k = 2$. The first square is made by the 2^k factorial design and the second one is added by CCD. The result is depicted on the left-hand side. See also Figure 3.20 in [NIS16].

tered(CCF) are defined. As mentioned before, the factorial design covers the minimum and maximum values of the numeric option range. Thus, it may not be possible to select values which are smaller than the lower bound or higher than the upper bound. To overcome this problem, the star points as well as the factorial points are scaled down to the actual bounds by using the value of α and the distance from the center to the factorial points. This type of Central Composite design is called CCI. As CCI needs at least 5 different values in the numeric range, which is also not always possible, the star points were placed on the center of every edge of the rectangle and resulted in the CCF design.

In SPL Conqueror CCI was implemented and thus, we do not investigate the other types. Further information on the different types of CCD is provided in [NIS16]. The selection of numeric option values using CCI for two numeric configuration options is shown in Figure 2.4. In the figure, the cells with red content depict configurations that are ignored, whereas green-colored cells with a point in the center are chosen configurations.

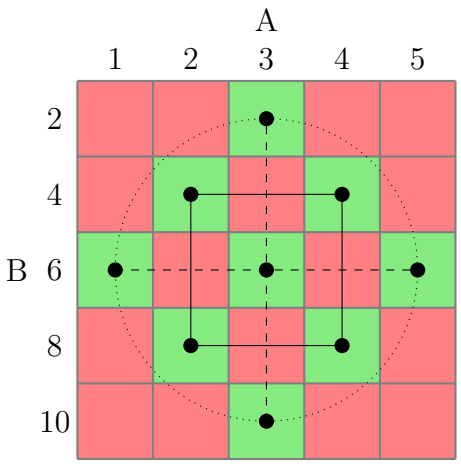

Figure 2.4: Selection of numeric values with the Central Composite Design using CCI and two numeric options A and B where $A = [1..5; 1]$ and $B = [2..10; 2]$.

2.3 Similarity and Distance Measures

Distance and similarity measures are used to determine how close two elements of the same kind are to each other [XX11]. Since we want to express the similarity or dissimilarity of performance-influence models with a numerical value, we use distance and similarity measures in this work. When using a distance measure, the result of two closer instances are smaller than the result of two distant instances. In contrast, a similarity measure results in higher values the closer both instances are to each other. Moreover, the value of an similarity measure is in between 0 and 1, where 0 means absolutely no similarity and 1 means that the instances are the same. However, this does not inherently apply to distance measures.

Besides, the distance and similarity measures can be applied to different input types, such as *Point Data*, where each point in the n-dimensional space is a multidimensional vector of length n and where the i-th number is the coordinate in the i-th dimension [DB79]. There are also other input types than the one presented before. However, these input types are not described in more detail, as they are not relevant in this work. Additionally, there are multiple axioms that can be satisfied by the distance and similarity measures. These are presented in the following.

Let $\mathcal{S}$ denote the set of all points in an n-dimensional space, d, s the distance and the similarity measures with $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ and $s : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$.

Then, according to the work of Xu et al. [XX11], for distance measures the satisfiable conditions are:

1. $\forall x, y \in \mathcal{S} : d(x, y) = 0 \Leftrightarrow x = y$ Identity of Indiscernibles
2. $\forall x \neq y \in \mathcal{S} : d(x, y) \geq 0$ Non-Negativity or separation axiom
3. $\forall x, y \in \mathcal{S} : d(x, y) = d(y, x)$ Symmetry
4. $\forall x, y, z \in \mathcal{S} : d(x, z) \leq d(x, y) + d(y, z)$ Triangle Inequality

According to the work of Xu et al. [XX11], the satisfiable conditions for the similarity measures are as follows:

1. $\forall x \neq y \in \mathcal{S} : s(x, y) \geq 0$ Non-Negativity
2. $\forall x, y \in \mathcal{S} : s(x, y) = 1 \Leftrightarrow x = y$ Identity of Indiscernibles
3. $\forall x, y \in \mathcal{S} : s(x, y) = s(y, x)$ Symmetry
4. $\forall x, y, z \in \mathcal{S} : s(x, z) \geq s(x, y) + s(y, z)$ Triangle Inequality

Regarding the satisfied conditions, these measures can be categorized in multiple types of metrics [GV02], which are:

- *Metric*: A *metric* has to satisfy every property that was mentioned above, namely *Non-Negativity*, *Identity of Indiscernibles*, *Symmetry* and *Triangle Inequality*.

- *Pseudo-Metric*: These are functions that satisfy *Identity of Indiscernibles*, *Symmetry* and *Trianlge Inequality*, but no *Non-Negativity*.
- *Semi-Metric*: For this kind of measure, every property but the *Triangle Inequality* holds.
- *Semi-Pseudo-Metric*: This kind of measure is a combination of *Semi-Metric* and *Pseudo-Metric*. So, *Non-Negativity* and *Triangle Inequality* is not satisfied by this kind of metric.

This categorization applies to distance functions as well as to similarity measures. Furthermore, there are also additional axioms that allow other combinations. However, as they are not relevant for the comprehension performed in this thesis, they are not further discussed.
Although there are many different distance and similarity measures, some of which are presented by Cha et al. [Cha07], we focus on the *Cosine Similarity*, *Jaccard* and the *Minkowski Distance*, which we present in the following.

2.3.1 Cosine Similarity

The *Cosine Similarity* [NB10] measures the cosine of the angle between two vectors in an n-dimensional space. The formula of the cosine similarity for two n-dimensional vectors a and b is as follows:

$$cos(a,b) = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Like the cosine function, the resulting similarity ranges between $[-1,1]$, so the non-negativity property is not given. Moreover, the triangle inequality does also not hold, which makes this measure a semi-pseudo-metric [Kry14].

2.3.2 Jaccard

Another similarity measure is the *Jaccard Index*, which is also known as the *Jaccard Similarity* [HHH+89]. This measure is used to express the similarity of two different sets. However, Jaccard can additionally be used for estimating the similarity of two multidimensional vectors. For two n-dimensional vectors a and b, the formula is:

$$J(a,b) = \frac{\sum_{i=1}^{n} \min(a_i, b_i)}{\sum_{i=1}^{n} \max(a_i, b_i)}$$

Since the non-negativity does not hold for negative vectors and the triangle inequality does it neither, this is also a semi-pseudo-metric.

2.3.3 Minkowski

One of the most popular distance measures is the Minkowski distance of order p [Cha07]. The general formula for two multidimensional vectors is:

$$M_p(a,b) = (\sum_{i=1}^{n} |a_i - b_i|^p)^{\frac{1}{p}}$$

According to Hardy et al. [HLP52], the Minkowski distance is a distance metric for the $p \geq 1$. In this thesis, we use different forms of the Minkowski distance, namely the Manhattan distance ($p = 1$), the Euclidean distance ($p = 2$), Minkowski distance of order 3 and Chebyshev ($\lim_{p\to\infty}$).

2.4 Multigrid

In this section, the multigrid approach is presented, because both case study systems are multigrid systems. Multigrid is the most efficient way to solve partial differential equations (PDEs). These PDEs describe physical or chemical processes, such as movement of waves as described by LeVeque et al. [LeV97]. Another possibility to solve a PDE is a direct solver like the gaussian solver, which solves the PDE directly achieving high accuracy but also a high runtime ($O(N^3)$, where N denotes the number of unknowns in the PDE). In contrast to the direct solver, the multigrid approach is an iterative solver that has a low runtime ($O(N \log \epsilon)$, where N is the number of unknowns and ϵ is a constant value). In the following, we provide an overview of the multigrid approach and describe its components in detail.

2.4.1 Overview

In this section, we provide an overview of the multigrid approach, which is depicted in Figure 2.5. Before the application of the multigrid approach, the PDE has to be discretized to obtain an equivalent grid, which we describe in Section 2.4.2. Each data point on the grid represents one unknown in the original PDE. As a direct solver cannot be applied to the grid because of the high runtime and the memory consumption, it has to be brought into a form that is easier to solve. To do so, a hierarchy of grids is used to reduce the size of the grid before applying the solver. Therefore, a restriction operation is performed to transfer the grid from a fine to a coarser grid and hence, the grid contains less unknowns than before. We describe the transfer from a fine to a coarser grid in Section 2.4.3. Afterwards, a smoother is used to turn high-frequent errors into low-frequent errors, since a high-frequent error cannot be mapped by executing the restriction. We explain the smoothing process in Section 2.4.4. This process is repeated until the coarsest level is reached. Then, the solver is executed to obtain a solution. The solver is presented in Section 2.4.5. Now, the grid has to be brought to the initial size. Thus, the prolongation and additional smoothing is performed, which updates the values of the finer grids. This process is also shown in Figure 2.5. Besides, the V-cycle is repeated until the solution error is appropriate.

Performing such a cycle is also named iteration. So, performing 5 iterations with a V-cycle means that the V-cycle is repeated 5 times to achieve a smaller error regarding the solution. Other frequently used cycles are the W-cycle and the F-cycle as presented in [TS01].

2.4.2 Grid

The grid is the most basic compound of the multigrid approach and is built by discretizing the PDE. That means that the PDE being considered is mapped to the

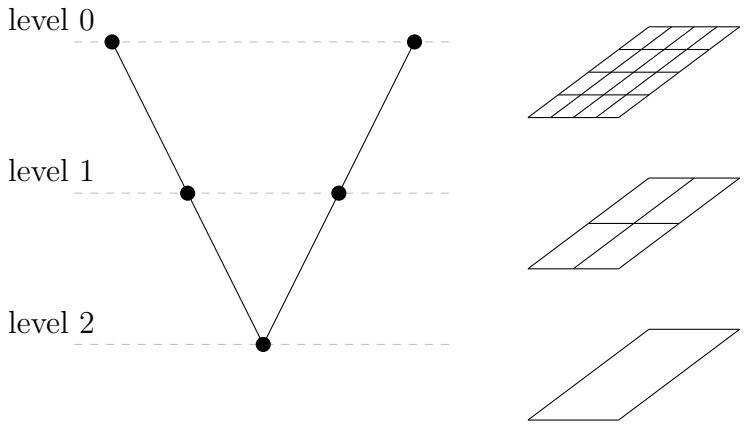

Figure 2.5: A V-cycle beginning on level 0 where the initial grid is of the size 4×4. On the coarsest level, level 2, the grid is of the size 1×1. On the right-hand side, the grids of each level are shown.

grid. To do so, often the finite element and the finite difference methods are used to discretize the PDE [BS07]. In this work, the process of mapping the equations onto a grid is not explained in more detail, as the measured software creates the grid automatically. Every crossing of a horizontal and a vertical line stands for one unknown.

Additionally, the size of the grid in each dimension can oftenly be configured. In our work, the grid size is denoted as nx, ny and nz for the grid size in each dimension. Although grids can be of different shapes, we only focus on rectangular grids in this work.

2.4.3 Restriction and Prolongation

The restriction and the prolongation are operations that are performed to transfer information from one grid to the other. As mentioned previously, the restriction coarsens the grid, so that only a subset of the nodes are considered in the next steps. In more detail, the grid is separated into 3×3 grids and then mapped on a single block (see Figure 2.6). Thereby, different coarsening operators can be used, which differ in runtime, parallelism and efficiency property.

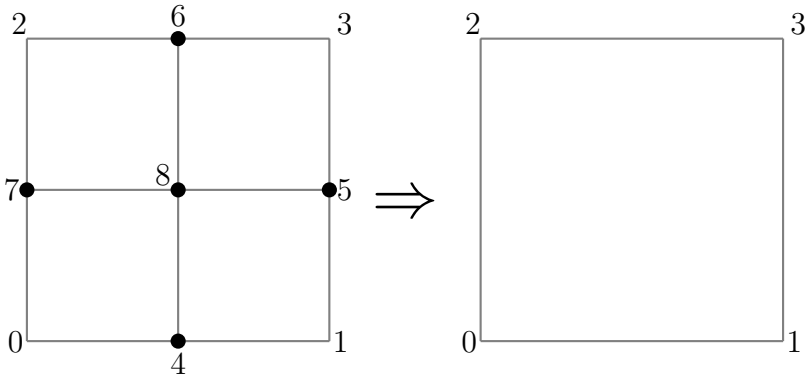

Figure 2.6: The restriction in more detail by restricting a 2×2 grid to a 1×1 grid.

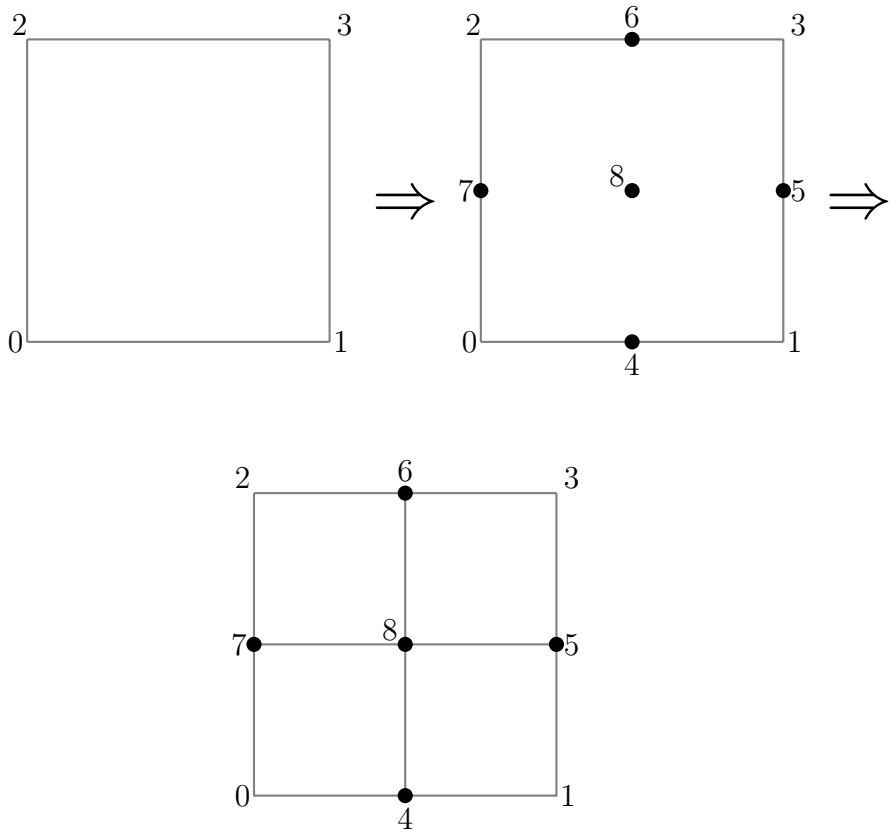

Figure 2.7: Prolongation from a single grid cell to a 2×2 grid.

In comparison to that, the prolongation is the inverse to the restriction. While the restriction transfers the information on a coarser grid, the prolongation updates the information on the finer grid. The input of the prolongation is an $n \times n$ grid. Afterwards, on every cell of the grid a prolongation operation is performed according to Figure 2.7. In this figure we show the extraction of the information of every cell and the transfer of the single cell to the corresponding 2×2 cells on the finer grid. Afterwards, we consider the finer $2n \times 2n$ grid instead of the $n \times n$ grid after the prolongation step.

2.4.4 Smoothing

Because of the restriction, a pre-existing error could be amplified, since the coarsening turns low-frequent error into high-frequent error. A low-frequent error means that the error of a grid point does not differ too much from the neighboring grid points. By contrast, a high-frequent error means the opposite. Since high-frequent error can not be represented on a coarser grid, the high-frequent error has to be turned to a low-frequent error. To convert high-frequent error into low-frequent error, a smoother can be used multiple times as depicted in Figure 2.8. The application of the smoother after the restriction is called pre-smoothing. High-frequent errors does also appear after the prolongation. Therefore, smoothing can be applied additionally after prolongation, which is called post-smoothing. Note that, although the error is converted using a smoother, the error is not necessarily decreased. In

literature, there are different smoothers that can be applied. Famous examples are the Gaussian elimination and the Jacobi elimination [TS01].

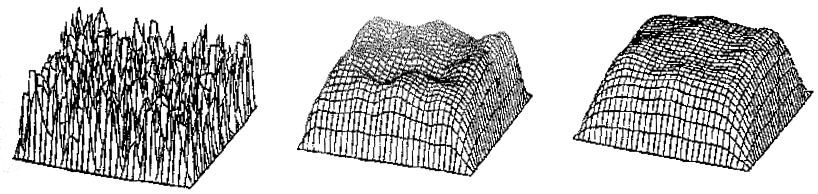

Figure 2.8: The error without smoother (left-hand side), the error with 5 smoother repetitions (midle) and with 10 repetitions (right-hand side) [TS01].

2.4.5 Solver

In the multigrid approach, the solver is applied to the coarsest grid to solve the PDE. Applying a multigrid solver, such as the gaussian elimination, on a grid has the highest complexity over all multigrid components. For instance, the gaussian elimination has a complexity of $\mathcal{O}(n^3)$, where n is the number of unknowns. Instead of applying the multigrid solver directly to the grid, the multigrid solver is applied to the coarsest grid with only a few unknowns to save execution time. However, applying the solver to the coarsest and not the finest grid produces an error, which makes it necessary to repeat the whole multigrid cycle.

3. Case Studies

In this chapter, we give an introduction to SMG2000 in Section 3.1 and afterwards to BoomerAMG in Section 3.2. For both systems, we describe the functionality of the programs as well as the corresponding analytical performance models.

3.1 SMG2000

In this section, we present the multigrid program SMG2000 [Car01], which is part of the *Hypre* library [FY02]. Firstly, we explain the functionality of SMG2000 in Section 3.1.1 and explain the specific characteristics of the program in relation to the multigrid approach. Afterwards, we present the analytical performance models of SMG2000 and discuss it in Section 3.1.2.

3.1.1 Functionality

SMG2000 is an implementation of a semi-coarsening approach. Semi-coarsening minds situations where the deviations of neighboring cells are high. In this case, the semi-coarsening maps them as two different cells on the coarser grid, whereas in standard coarsening such situations are ignored [dZ96]. For this system, analytical performance models were created by Brown et al. [BFJ00]. In their models, they consider the prediction of the relaxation process, which needs a major part of the runtime (about 90%).

The processor topology and the grid size per block on the different dimensions are the numeric options that are observed in this work and aspects that are included in the analytical performance model. The grid allocation is shown in Figure 3.1, where each block is a subgrid of size $nx \times ny \times nz$. Hence, the higher the number of total processes $p_{total} = p_x \cdot p_y \cdot p_z$, the greater the grid size.

The analytical performance models predict the performance when performing SMG2000 on grids with 1 to 3 dimensions, one performance model for each dimension. In the following section, we present the analytical performance model of SMG2000 in more detail.

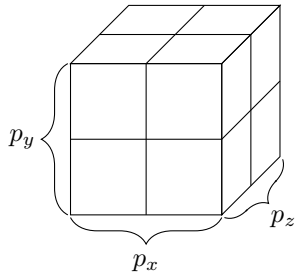

Figure 3.1: A grid on a multicore system. Every cell is a subgrid of size $nx \times ny \times nz$ and is processed by one core on the computing machine.

3.1.2 Analytical Performance Models

The analytical performance model of SMG2000 by Brown et al. [BFJ00] considers only the relaxation phase of the respective multigrid approach. As it turned out, 90% of the runtime is needed by the relaxation, which constitutes a major part of the runtime. In SMG2000 the grid can have different dimensions, namely from 1 to 3. Accordingly, Brown et al. have created an equation for each dimension with increasing complexity.

Since SMG2000 uses the *Message Passing Interface* (*MPI*) for communication between multiple processes, the latency as well as the bandwidth are crucial factors in the analytical performance model. In the model it is assumed that the time to deliver a number of values of the type `double`, called n, from one process to another is:

$$T_{send} = \alpha + \beta \cdot n$$

In the equation, α depicts the latency of *MPI* messages and β the inverse bandwidth of a `double` value sent via *MPI*.
The equation for SMG2000 running on a 1-dimensional grid is the following:

$$T^{1D} = 2 \cdot L_x \cdot \alpha + 2 \cdot L_x \cdot \beta + 6 \cdot N \cdot f$$

where T is the runtime in microseconds, $L_i = \log_2(p_i \cdot N)$, N denotes the size of the grid in one dimension, and f is the inverse of floating point operations per second. Furthermore, α, β, and f are constants that depend on the system and can only be resolved by benchmarks on the respective systems. When investigating the equation, the operations performed in the relaxation phase are considered by the last term $6 \cdot N \cdot f$, whereas the message exchange in the relaxation phase is represented by $2 \cdot L_x \cdot \alpha + 2 \cdot L_x \cdot \beta$.

However, this equation is currently not related to the configuration options, which we have mentioned in the previous section. Therefore, the variables are mapped to their corresponding names of the configuration options. As this equation is meant for 1-dimensional grids, we strictly replace N by nx. The outcome of the replacement upon the equation is shown below.

$$T^{1D} = 2 \cdot \log_2(px \cdot nx) \cdot \alpha + 2 \cdot \log_2(px \cdot nx) \cdot \beta + 6 \cdot nx \cdot f$$

Additionally, in the logarithmic function appear multiple unknowns. Since it holds that $\log(a \cdot b) = \log(a) + \log(b)$, the logarithmic function can be split into two terms. This is necessary, as the equation of the analytical performance models and the empirical performance-influence models have to be converted in a similar form to make comparisons of both performance models easier. Moreover, we have configured SPL Conqueror in such a way that only one option appears in the logarithm function, as it would lead to ambiguity otherwise. The outcome after splitting the logarithmic functions is the following:

$$T^{1D} = 2 \cdot \log_2(px) \cdot \alpha + 2 \cdot \log_2(nx) \cdot \alpha + 2 \cdot \log_2(px) \cdot \beta + 2 \cdot \log_2(nx) \cdot \beta + 6 \cdot nx \cdot f$$

This procedure is repeated for the analytical performance model for 2 and 3 dimensions. Since these conversions behave similar for the all dimensions, we consider only the initial and the converted equations.

The model for a 2-dimensional grid is as follows:

$$T^{2D} = 4 \cdot L_y(L_x + 1) \cdot \alpha + 4 \cdot N \cdot (L_x + L_y) \cdot \beta + 20 \cdot N^2 \cdot f$$

Furthermore, in the paper $nx = ny$ holds for two-dimensional grids. Hence, every appearance of N is replaced by nx without loss of generality. The conversion yields:

$$\begin{aligned} T^{2D} = {} & 4 \cdot \log_2(py) \cdot \log_2(px) \cdot \alpha + 4 \cdot \log_2(py) \cdot \log_2(nx) \cdot \alpha + 4 \cdot \log_2(nx) \cdot \log_2(px) \cdot \alpha \\ & + 4 \cdot \log_2(nx) \cdot \log_2(nx) \cdot \alpha + 4 \cdot \log_2(py) \cdot \alpha + 4 \cdot \log_2(nx) \cdot \alpha + 4 \cdot nx \cdot \log_2(nx)\beta \\ & + 4 \cdot nx \cdot \log_2(px)\beta + 4 \cdot nx \cdot \log_2(nx)\beta + 4 \cdot nx \cdot \log_2(py)\beta + 20 \cdot nx \cdot nx \cdot f \end{aligned}$$

In the following, we present the analytical performance model for a 3-dimensional grid:

$$T^{3D} = 4 \cdot L_z \cdot (1 + 2L_y(L_x + 1)) \cdot \alpha + 4 \cdot N^2 \cdot (L_z + 2L_x + 2L_y) \cdot \beta + 48 \cdot N^3 \cdot f$$

As before, T symbolizes the runtime in microseconds, $L_i = \log_2(p_i \cdot N)$, N denotes the size of the grid in one dimension. According to the previous equations, the same rules are applied to these transformations. Furthermore, $nx = ny = nz$ holds in this equation with respect to Brown et al., so we replace N again by nx. The result is:

$$\begin{aligned} T^{3D} = {} & 4 \cdot \log_2(pz) \cdot \alpha + 8 \cdot \log_2(nx) \cdot \alpha + 8 \cdot \log_2(pz) \cdot \log_2(py) \cdot \log_2(px) \cdot \alpha \\ & + 8 \cdot \log_2(pz) \cdot \log_2(py) \cdot \log_2(nx) \cdot \alpha + 8 \cdot \log_2(pz) \cdot \log_2(nx) \cdot \log_2(px) \cdot \alpha \\ & + 8 \cdot \log_2(pz) \cdot \log_2(nx)^2 \cdot \alpha + 8 \log_2(nx) \cdot \log_2(py) \cdot \log_2(px) \cdot \alpha \\ & + 8 \cdot \log_2(nx)^2 \cdot \log_2(py) \cdot \alpha + 8 \cdot \log_2(nx)^3 \cdot \alpha \\ & + 8 \cdot \log_2(pz) \cdot \log_2(py) \cdot \alpha + 8 \cdot \log_2(pz) \cdot \log_2(nx) \cdot \alpha \\ & + 8 \cdot \log_2(nx) \cdot \log_2(py) \cdot \alpha + 8 \cdot \log_2(nx)^2 \cdot \alpha \\ & + 4 \cdot nx^2 \cdot \log_2(pz) \cdot \beta + 4 \cdot nx^2 \cdot \log_2(nx) \cdot \beta + 8 \cdot nx^2 \cdot \log_2(px) \\ & + 8 \cdot nx^2 \cdot \log_2(nx) + 8 \cdot nx^2 \cdot \log_2(py) + 8 \cdot nx^2 \cdot \log_2(nx) \\ & + 48 \cdot nx^3 \cdot f \end{aligned}$$

By viewing the general equations of the analytical performance model, they turn out to be easy to understand. As already mentioned, this could lead to problems, as in a simple equation too few influence factors are selected and hence, such performance models contain a higher prediction error.

3.2 BoomerAMG

In the following, we present the multigrid system BoomerAMG [Yan02], which is also part of the *Hypre* library [FY02]. First of all, we describe the characteristics of BoomerAMG in Section 3.2.1. Then, in Section 3.2.2, we present the analytical performance model which was proposed by Gahvari et al. [GBS+11].

3.2.1 Functionality

The Algebraic MultiGrid (AMG) [RS87] is a multigrid approach that is focused on solving extremely large, unstructured grids. An example of an unstructured grid is given in Figure 3.2, where the grid has no rectangular form as we have shown in Section 2.4.2. Since the grid has no rectangular form, a lot of optimizations cannot be applied. For instance, Chen et al. [CLB03] used a non-rectangular grid for the representation of an ocean model. The absence of specific shape properties makes the computation of AMG difficult to parallelize. However, with a parallel execution of unstructured grids on multiple cores, the overall runtime of this approach would be shortened. To parallize this, an algorithm was proposed by Cleary et al. [CFJ98], later implemented by Yang et al. [Yan02] and called BoomerAMG.

As done in SMG2000, we vary the processor topology px, py, pz as well as the grid size nx, ny, nz in each dimension. In this system, they use the same topology for the processors as in SMG2000. For instance, $px = 2$, $py = 1$, $pz = 3$ results in $p_{total} = 2 \cdot 1 \cdot 3 = 6$ processes. Every process sustains a subgrid of the size $nx \times ny \times nz$. The stepsize and the minimum and maximum value will be adjusted according to the system specifications of the executed system in Chapter 5.

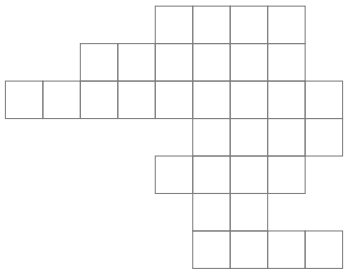

Figure 3.2: An example of an unstructured grid. In comparison to a structured grid, this type of grid has no special form.

Furthermore, multiple simulations use BoomerAMG such as groundwater flow [KH14] and electromagnetic flow [BDGGW05]. Hence, performance models are crucial for this application for estimating the runtime of a given configuration. In the work of Gahvari et al. [GBS+11], an analytical performance model is proposed, which we describe in the following section.

3.2.2 Analytical Performance Model

In this section, we present the analytical performance model for the algebraic multigrid system BoomerAMG that was proposed by Gahvari et al. [GBS+11]. In the work of Gahvari et al., a general analytical performance model is proposed, which is

called the baseline model. Additionally, the equation was refined by adding different penalties regarding the distance between nodes, the bandwidth and the parallelization. Since our measurement environment is rather small in comparison to the different clusters used by Gahvari et al. [GBS+11] for the evaluation, we consider only the baseline model. In contrast to the model of SMG, they consider more complex parameters, which have to be obtained by instrumentation. To this end, they use the TAU Performance System [SM06] to get the number and size of sent messages via MPI. In Chapter 5, we explain TAU in more detail. In the following, let P be the total number of processes in a configuration, C_i denotes the number of unknowns on grid level i, $s_i, \hat{s}_i$ are the average number of non-zeros per row of the grid in the solve and the interpolation phase, respectively, $p_i, \hat{p}_i$ denote the global maximum number of *MPI_Send* and *MPI_Isend* operations on level i for the solve and interpolation step, $n_i, \hat{n}_i$ symbolize the maximum number of elements sent with *MPI_Send* or *MPI_Isend* on level i in the solve and the interpolation phase. Furthermore, t_i stands for the time per floating point operation on level i, α for the latency of *MPI* messages and β for the inverse bandwidth of *MPI* per double.

Again, the analytical performance model assumes that the cost of sending an *MPI* message with n elements is:

$$T_{send} = \alpha + n\beta$$

Based on this assumption, the overall time of an BoomerAMG execution with G levels using a V-cycle is modeled as:

$$T_{solve}^{AMG} = \sum_{i=0}^{G} T_{solve}^{i}$$

In comparison to the analytical performance model of SMG2000, this analytical performance model is more detailed in that every level in the V-cycle is modeled:

$$T_{solve}^{i} = T_{smooth}^{i} + T_{restrict}^{i} + T_{interp}^{i}$$

$$T_{smooth}^{i}(\alpha, \beta) = 6\frac{C_i}{P}s_i t_i + 3(p_i\alpha + n_i\beta)$$

$$T_{restrict}^{i}(\alpha, \beta) = \begin{cases} 2\frac{C_{i+1}}{P}\hat{s}_i t_i + \hat{p}_i\alpha + \hat{n}_i\beta & \text{if } i < G \\ 0 & \text{if } i = G \end{cases}$$

$$T_{interp}^{i}(\alpha, \beta) = \begin{cases} 0 & \text{if } i = 0 \\ 2\frac{C_{i-1}}{P}\hat{s}_i t_i + \hat{p}_i\alpha + \hat{n}_i\beta & \text{if } i > 0 \end{cases}$$

Additionally, one model was created for different components, namely smoothing, restriction and interpolation. This model is called the *Baseline Model* and does not consider any further delays which may occur.

To sum it up, the analytical performance model proposed by Gahvari et al. [GBS+11] is more complex than the one from SMG2000. Moreover, this analytical performance model needs further information from the software run that can be obtained by the instrumentation with TAU.

4. Analyzation

Until now, the analytical performance models and the empirical performance-influence models were described separately. In the following, we describe multiple ways to compare the different models. This can be categorized by the type of information we use for the comparison, namely in the *syntactical* and the *semantical* comparison strategy. The first approach, the syntactical comparison, uses only syntactical information and is presented in Section 4.1. Finally, in the semantical comparison, the performance-models are evaluated using their results as we describe in Section 4.2. The interpretation of the results of both comparison strategies is done in Chapter 5.

4.1 Syntactical Comparison

Since we want to compare all terms with the same predicted influences, we compare the performance models by creating so-called equivalence classes regarding the appearing variables in a term. For instance, $2 \cdot x + 0.5 \cdot x^2$ are both terms in the equivalence class x, as they both describe the influence of the configuration option x. Furthermore, x has a linear influence on the performance of factor 2 and a quadratic influence of factor 0.5. In the following, we call the influence factor the value of a term. Assume, we have the following models:

$$T_{analytical} = 3 \cdot x + 0.4 \cdot y + 1 \cdot y^2 + 2 \cdot \log_2(x) \cdot y + 1 \cdot x \cdot y$$

$$T_{empirical} = 5 \cdot x + 0.5 \cdot x^2 + 3 \cdot y + 1 \cdot y^2$$

where x, y are numeric options. Firstly, we sort the terms of each model in different equivalence classes. In the first model, we categorize the terms in the following equivalence classes:

- x: $3 \cdot x$
- y: $0.4 \cdot y + 1 \cdot y^2$

- $x; y$: $2 \cdot \log_2(x) \cdot y + 1 \cdot x \cdot y$

The equivalence classes of the second performance models are:

- x: $5 \cdot x + 0.5 \cdot x^2$
- y: $3 \cdot y + 1 \cdot y^2$

Now, if we want to compare both models, we compare each term of an equivalence class of one performance model with each term of the same equivalence class of the other performance model. To do so, we apply the following formula on each term of both performance models:

$$scoreOfATerm = \begin{cases} -2 & \text{if the other model has no such equivalence class} \\ -1 & \text{if the equivalence class exists but no such term} \\ 1 + simValue & \text{if the term exists in the other model with a different value} \\ 2 & \text{if the term exists in the other model with the same value} \end{cases}$$

The $simValue$ is defined as $\max(0, 1 - \frac{|e-a|}{a})$, where e is the influence of the term from the empirical performance-influence model and a is the influence of the term from the analytical performance model. According to the formula above, we reward a term that is present in both performance models with a positive score, whereas the absence of the term or of the equivalence class results in negative score. The penalty has to result in a negative score, since otherwise a performance model considering all possible configuration options and interactions would achieve a high score. In the case that a term is present in both models but with different influences, we compute the similarity of both influences. Therefore, we have used the formula of the error rate $\frac{|e-a|}{a}$ and inversed its result. Since we want that this reward is between the interval $[1, 2]$ and the values of the error rate can be above 1, we take the maximum of 0 and the inverse error rate. These scores are summed up and build the general score for the syntactical comparison of both models. The above formula is applied to every kind of influence function (e.g., logarithmic, quadratic) of a variable only once. Influence functions that do not appear at all are not considered. In our example, the score for x in the equivalence class x is only computed once. Thus, for the equivalence class x, the result would be $(1 + (1 - \frac{|5-3|}{3})) + (-1) = 0.33$, as x is present in both equivalence classes but x^2 only in one. Note that we always view every term of both equivalence classes. The result for the second equivalence class, y, is $1 + 2 = 3$, as y appears in both classes having a different value, whereas y^2 is completely the same and thus is awarded with the highest score, 2. However, the lowest score is achieved by the equivalence class $x; y$, where $-2 + (-2) = -4$ is the outcome, as no single term was found by the second model and thus, both terms get the score -2. The overall score is then the sum of the previous results, $0.33 + 3 + (-4) = -0.66$.

4.2 Semantical Comparison

In the previous section, we have only included syntactical information in the comparison of the models. Now, we include only semantical information consisting of the information about numeric options as well as the measured configurations. To

compare the models with the real results as well as with each other, the outcome of the performance model is computed for each configuration. In our example, the option x has the minimum value 1, the maximum value 100 and a step size of 1. Furthermore, y has a minimum value of 0, a maximum value of 50 and 2 as step size.

The results of both models are presented in Figure 4.1 and show that the models behave differently. Since we have created both performance models for exemplary use, we can not show real results of measurements at this time.

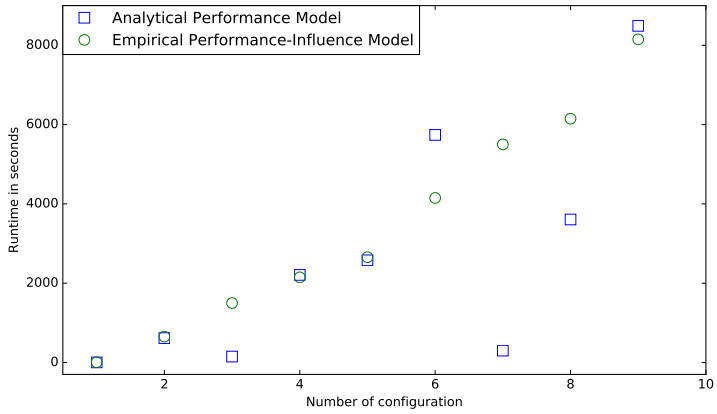

Figure 4.1: The visualization of the results of some configurations of the exemplary performance models. The x-axis is the number of configurations, which are sorted in ascending order regarding the result of the first performance model.

Next, we apply the distance and similarity measures on each of these configurations to calculate the difference between these models. As a reference, we have calculated the relative error rate by $\frac{|measured-predicted|}{measured}$ denoted as *Error Rate* and use this information to investigate if some distance or similarity measures provide similar or even more useful information. The outcome is presented in Table 4.1 and shows that the distance and similarity measures behave very differently, which will be discussed in Chapter 5.

Table 4.1: The semantical comparison of both performance models. This shows the results of the different distance and similarity measures. Note that these values can not be used to form statements, as the underlying performance models were created for the exemplary illustration.

Name	Result
Error Rate	36.1%
Cosine Similarity	87.8%
Jaccard	65.9%
Manhattan Distance	1244
Euclidean Distance	4219132
Minkowski Distance	18170166746
Chebyshev	5200

5. Evaluation

After providing relevant knowledge in Chapter 2 and Chapter 3, we are now able to compare the analytical performance models and empirical performance-influence models. Therefore, we describe the setup of our measurements in Section 5.1. Based on the measurements and on the strategies presented in Chapter 4, we perform a comparison of the models for SMG2000 in Section 5.2.1 and for BoomerAMG in Section 5.2.2.

5.1 Setup

The underlying measurements for this thesis were performed on one cluster, which we describe in Section 5.1.1. In Section 5.1.2, we provide information about the software used for obtaining the measurements.

5.1.1 Cluster

We executed both case studies on a cluster which consists of normal workstations, but pointed out that the analytical performance models are created with the consens of higher parallelization than on one quad core CPU. To this end, we performed all measurements, which are needed for the creation of the empirical performance-influence model on a cluster, where more parallelization is possible, namely `Chimaira`.

Hardware Specification. The cluster *Chimaira* consists of 17 nodes, each consisting of an `Intel Core i7-4770 @ 3.40 GHz` with 4 CPU cores, 8 hyperthreads and 32 GB RAM. To minimize the latency and to increase the bandwidth when messages are sent between these nodes, Chimaira has an additional 10Gb network interface. The nodes are all connected to this switch, whereas only internal messages are exchanged. Every other traffic is sent over another network switch, where the latency is consequently unpredictable. Moreover, on every node of the cluster the Linux operating system `Ubuntu 16.04.1 LTS` is installed.

Since measurement bias could affect the performance of the software, we met additional precautions. To ensure that every process is executed by one core, we turned

off hyperthreading, because the behavior of hyperthreads is unpredictable and also leads to higher deviations in the performance results.

Moreover, we have repeated each measurement at least 3 times and have also computed the standard deviation of each configuration. The standard deviation of each configuration was less than 10%.

5.1.2 Software

To determine all unknowns in the analytical performance models, we use different software that we present in the following.

Benchmark Systems.

HPCC. The High Performance Computing Challenge (HPCC) is a benchmark that can be used to determine performance characteristics of a system. To this end, the performance of matrix multiplication, parallel matrix transpositions and bandwidth/latency tests are included in this benchmark. In our work, we use the test `b_eff` to compute the overall latency denoted as α and inverse bandwidth β of the analytical performance models described in Section 3.1.2 and Section 3.2.2. Therefore, we have used version `1.5.0` of the benchmark.

MVAPICH. Another benchmark for the latency and bandwidth of MPI messages is MVAPICH [Tea02], which is based on MPICH, an implementation of MPI. In our work, we have used MVAPICH to validate the results obtained by the HPCC benchmark `b_eff`. In addition to HPCC, MVAPICH lists the bandwidth of diverse numbers and sizes of messages. On our clusters, we have used the version `MVAPICH2 2.2`.

Intel® LINPACK. LINPACK [DLP03] is another benchmark, which is used for measuring the double precision floating point operations per second (FLOPS). It is also used for the global TOP500 list of supercomputers. For our purpose, we have used LINPACK version `2017.1.013`.

SpMV. The Sparse Matrix-Vector multiply (SpMV) [Gah06] is a benchmark for matrix-vector multiplication. Since different factors such as the density of non-zero entries in the matrix and its dimension have an impact on the performance of such operations, it is important to measure the different configurations of a matrix-vector multiplication.

Code Instrumentation.

TAU. As we mentioned in Section 3.2.2, information, such as the number of messages sent from one node, is used in the analytical performance model of BoomerAMG. To measure the information, we profile the number of sent messages and their respective. To this end, we have used the Tuning and Analysis Performance System® (TAU) [SM06]. This toolkit offers multiple ways of instrumenting the code of a certain software, namely *dynamic*, *source* and *selective* instrumentation, which differ in the set of information they provide.

Table 5.1: The ranges for the processor topology parameters for Chimaira. The first value in between square brackets is the minimum value, the second is the maximum value and the third stands for the step size of the configuration option.

Dimension	*px*	*py*	*pz*
1D	[1..4; 1]	1	1
2D	[1..4; 1]	[1..4; 1]	1
3D	[1..4; 1]	[1..4; 1]	[1..4; 1]

The source instrumentation is capable of profiling user-defined methods, such as the methods provided by the *Hypre* library. Therefore, the whole program has to be compiled with specific compilers provided by TAU. To this end, they provide different compilers for different source code languages, in more detail, they use `tau_cc`, `tau_cxx` and `tau_f90` for code written in C, C++ and Fortran, respectively. After the recompilation, each run of the software produces one *profile* file for each processor, which contains information about how much time was spent in user-defined and MPI methods as well as the number of messages and number of elements sent in each of this method. To obtain the information of every single level from the V-cycle explained in Section 2.4, we have modified the code by splitting each level in one separate method.

Additionally, TAU offers a tool, *pprof* for the textual representation of the instrumentation. For our evaluation, we have used *pprof* for extracting the desired information.

5.2 Obtaining Parameters for the Evaluation

In this section, we describe how we have obtained the parameters for the measurements of SMG2000 and BoomerAMG as well as for the analytical performance models. Firstly, we describe how we have measured the performance of SMG2000 in Section 5.2.1 and afterwards we do the same for BoomerAMG in Section 5.2.2. Moreover, we show the results of the benchmarks for the latency, bandwidth and floating point operations per second in Section 5.2.3.

5.2.1 SMG2000

To determine the minimum and maximum value as well as the step size, we investigate on every numeric configuration. On the one hand we have px, py and pz, which are the number of processors in each dimension. These parameters were chosen with a minimum value of 1, a maximum value of 4 and step size 1 since we were restricted in the overall number of processes. On the other hand, nx, ny and nz denote the grid size of every process in each dimension. The selected values of each of these parameters depends on the cluster and the dimension of the grid, which is presented in Table 5.2 for Chimaira. In the table, we use the notation from Section 2.2.2 for defining the ranges of the numeric options. The maximum value of each parameter was chosen in such a way that the main memory is not completely used, as it would produce `OutOfMemory` errors.

Table 5.2: The numeric option ranges of the different dimensions and the constraints for the cluster Chimaira.

Dimension	nx	ny	nz	Constraint
1D	[3 500 000..4 000 000; 1 000]	1	1	-
2D	[1 000..3 700; 10]	[1 000..3 700; 10]	1	$nx = ny$
3D	[50..220; 10]	[50..220; 10]	[50..220; 10]	$nx = ny = nz$

After determining the range of the numerical options, we have measured each configuration three times. For the evaluation, the standard deviation of every configuration was computed and is less than 10% for every measured configuration.

5.2.2 BoomerAMG

Firstly, the parameters for the processor topology, px, py and pz are chosen in the same way as in SMG2000, which is shown in Table 5.1. The only difference to SMG2000 is that we use at least 2 nodes on Chimaira, since the analytical performance model is focused on a high parallel execution according to Gahvari et al. [GBS+11]. Besides, the analytical performance model of BoomerAMG does not yield any constraints on the grid for each processor. The ranges of the numerical options in BoomerAMG are shown in Table 5.3.

Table 5.3: The numeric ranges for the subgrid sizes of BoomerAMG for Chimaira.

nx	ny	nz
[20..70; 10]	[20..70; 10]	[20..70; 10]

Obtaining the information of the analytical performance model of BoomerAMG was more difficult than SMG2000, as the analytical performance model needs specific values such as the number of messages sent by every node. Therefore, we have profiled the execution of each configuration with TAU. Since this has led to a much higher runtime, we have additionally executed every configuration without TAU. Hence, the executions with TAU instrumentation are only used to extract the information related to MPI, whereas the executions without TAU provides us with the performance information.

5.2.3 Benchmarks

The values computed in this section are used when the analytical performance models are evaluated. These values are the latency (α), the inverse bandwidth per value of type `double` (β) and the inverse double-precision floating point operations per μsec (f). To this end, we have used the software that is presented in Section 5.1.2. We have measured the latency and the bandwidth for different numbers of nodes on Chimaira. Since we use at least 1 and at most 7 nodes per run, we have measured the latency and bandwidth for 1-7. For our cluster, the number of nodes for every configuration is calculated firstly and afterwards, the corresponding values from the table are used. Moreover, we have used the values $\alpha = 0$ and $\beta = 0$ for every sequential run. The Table 5.4 shows the results and conversions of the latency and the bandwidth. Moreover, the value for Flops computed by Intel® LINPACK is 25.14 GFlops on Chimaira, which corresponds to 0.039 nsec/flop.

Table 5.4: The latency and bandwidth results on different amounts of processes on Chimaira. The latency was a direct result of the benchmark, whereas the inverse bandwidth has been calculated from the bandwidth.

#Nodes	Latency (nsec)	Bandwidth (MByte/s)	Inverse Bandwidth (nsec/double)
1	290	874	9.15
2	7 866	204	39.21
3	6 856	139	57.55
4	11 625	132	60.60
5	12 941	122	65.57
6	13 735	117	68.37
7	13 754	113	70.79

5.3 Learning Performance-Influence Models

Previously, in Section 2.2, we presented the tool SPL Conqueror for the creation of empirical performance-influence models. There, we have introduced different sampling heuristics for numerical options. Some of these sampling heuristics provide a parameter for specifying the number of elements in the learning set, namely the Plackett-Burman and the Random design. In the evaluation, we show the results of the different sampling heuristics in relation to the considered number of configurations. Another parameter is the *seed* in the Random design. For the evaluation, we have used the seeds 0 to 5 to minimize the influence of a generated empirical performance-influence model that achieves very good or very bad results. Another experimental design that allows different configurations is the Plackett-Burman design. As mentioned in Section 2.2.2, the design provides 6 different configurations. We have chosen 4 of these and depicted in Table 5.5.

Table 5.5: The chosen set of the Plackett-Burman design.

Measurements	Level
9	3
25	5
125	5
49	7

Moreover, because of the random nature of the Random design, we always calculate the mean value and the standard deviation of all seeds of a sample size. In the graphical representation of the results, we have always taken the empirical performance model with the lowest error rate with respect to the measurement results. We do this, as we want to show the best empirical performance-influence model that SPL Conqueror was able to learn with the sampling heuristics.

Besides, SPL Conqueror only learns logarithmic influence to the base 10, whereas the logarithmic functions in the analytical performance model of SMG2000 are to the base 2. However, we have made use of the logarithmic law $\log_a(x) = \frac{\log_b(x)}{log_b(a)}$ to convert the base and hence, this represents no restriction.

5.4 Comparison

In this section, we present the different comparison techniques described in Chapter 4 based on the multigrid systems SMG2000 and BoomerAMG. When performing the syntactical comparison of the performance models, we can choose between multiple values for α, β and f for computing the influence of a term. We observed that the selection of the value for these parameters has a low impact on the score of the syntactical comparison. To this end, we use the benchmark values for 4 nodes on *Chimaira*, as this is the median value. The results of the program are presented in Section 5.4.1 for SMG2000 and in Section 5.4.2 for BoomerAMG.

5.4.1 SMG2000

In this section, we investigate the comparison of the analytical performance models and the empirical performance-influence models for 1 to 3 dimensions of SMG2000. For each dimension, we perform the syntactical as well as the semantical comparison strategy.

1-Dimensional Grid.

For the 1-dimensional grid, Brown et al. [BFJ00] proposed the following analytical performance model:

$$T^{1D} = 2 \cdot log_2(px) \cdot \alpha + 2 \cdot log_2(nx) \cdot \alpha + 2 \cdot log_2(px) \cdot \beta + 2 \cdot log_2(nx) \cdot \beta + 6 \cdot nx \cdot f$$

Syntactical comparison. As a reminder, in the syntactical comparison the score according to the function presented in Section 4.1 between the analytical performance model and each empirical performance-influence model is computed. In the first row of Table 5.6, we present the achieved scores using different sampling heuristics for the generation of the empirical performance-influence models. The highest results were achieved by empirical performance-influence models using the Full Factorial and the Central Composite design. The syntactical score of both sampling heuristics was -4.2. The equation of the empirical performance-influence model generated with the Full Factorial design is as follows:

$$T^{1D}_{FF} = -6.023 - 0.023 \cdot px + 0.976 \cdot \log_{10}(nx) + 0.014 \cdot px^2$$

The equation produced with the Central Composite is:

$$T^{1D}_{CC} = -6.133 - 0.035 \cdot px + 0.994 \cdot \log_{10}(nx) + 0.016 \cdot px^2$$

Since the score of both empirical performance-influence models is almost the same, we observe the same equivalence classes and terms in both equations of the empirical performance-influence models. These equations have predicted the logarithmic but not the linear impact of nx. In contrast to the configuration option nx, the logarithmic impact of px was not found in the equations, but a linear influence of this option was found instead.

Table 5.6: The comparison of different sampling heuristic, namely Full Factorial (FF), Plackett-Burman (PB), Central Composite (CC) and Random design (R) with the analytical performance model. The measures were the Syntactical Comparison (SC), the Error Rate (ER), the Cosine Similarity (CS), the Jaccard Similarity (JS), the Manhattan Distance (MD), the Euclidean Distance (ED), the Minkowski Distance (MiD) and the Chebyshev Distance (CD). In the Random design, the standard deviation is included.

Measures	FF	PB(9,3)	PB(25,5)	PB(125,5)	PB(49,7)	CC	R(50)	R(125)
SC	−4.2	−7	−4.3	−4.3	−4.3	−4.2	−6.3(±14%)	−5.1(±18%)
ER	9.8%	9.4%	9.8%	9.8%	9.2%	10%	10.2%(±3%)	9.9%(±2%)
CS	99.3	99.4%	99.3%	99.3%	99.3%	99.2%	99.3%(±0.02%)	99.3%(±1.9%)
JS	90.3%	90.7%	90.2%	90.2%	90.2%	90%	90.1%(±0.17)%	90.2%(±0.1%)
MD	0.044	0.042	0.044	0.044	0.044	0.045	0.045(±1.5%)	0.044(±0.9%)
ED	0.0027	0.0023	0.0027	0.0027	0.0027	0.0029	0.0026(±2.3%)	0.0026(±5%)
MiD	0.0001	0.0001	0.0002	0.0002	0.0002	0.0002	0.00018(±6.6%)	0.00018(±11.5%)
CD	0.087	0.079	0.089	0.089	0.089	0.094	0.086(±4.1%)	0.086(±4%)

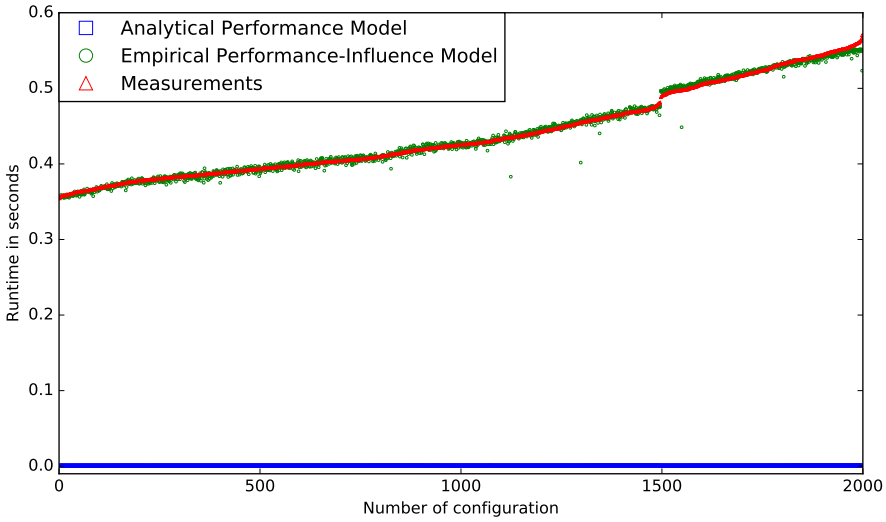

Figure 5.1: Results of the performance models in comparison to the execution time of the measurements in seconds. The configurations are sorted in ascending order according to the values obtained by the measurements.

Semantical comparison. The semantical comparison includes only semantical information by evaluating the performance models for every measured configuration. These information includes benchmark results shown previously in Section 5.2.3 and we show the results in Figure 5.1. In this figure, we use the empirical performance-influence model with the lowest error rate to the measurements, which was the empirical performance-influence model generated by using the Full Factorial design with 0.6%. Although we only show the empirical performance-influence model with the lowest error rate, the worst achieved error rate was 3.4% among all empirical performance-influence models. In contrast to this result, the analytical performance model has an error rate of 99%. After further investigation, we supposed that the benchmarked values were too low. In fact, as we change the values for the floating points operations per second, the analytical performance model produces better results. We observed that multiplying the latency, the inverse bandwidth and the inverse Flops by 500 results in a much better model. As soon as we encountered

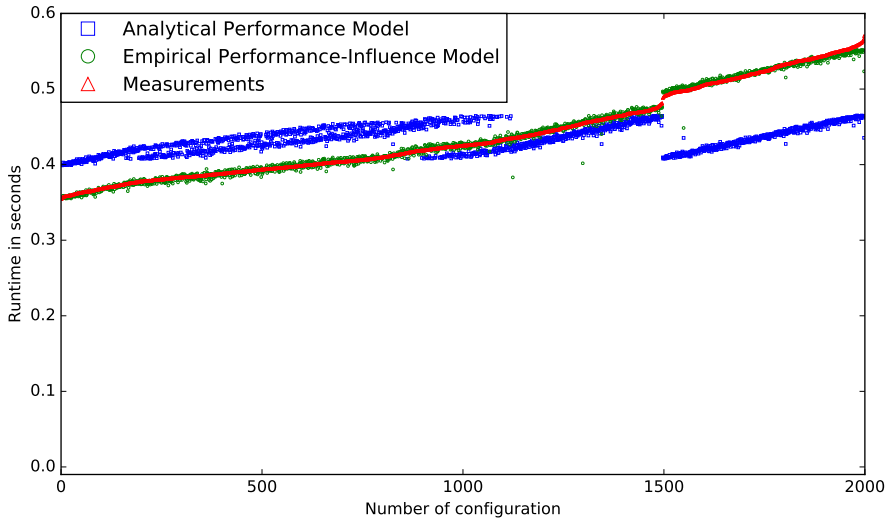

Figure 5.2: Results of the analytical performance model after adjusting the benchmark values.

Table 5.7: The results of the different measures when comparing the 1-dimensional analytical performance model results with the measurements.

Measures	Results
ER	9.7%
CS	99.3%
JS	90.3%
MD	0.044
ED	0.0027
MiD	0.0002
CD	0.13

this problem, we contacted the authors of Brown et al. [BFJ00]. Unfortunately, we have received no response at the time of writing. The outcome of this function is depicted in Figure 5.2 and has an error rate of 9.7%. Since we only want to compare performance models, we use the improved results of the analytical performance model in the following.

After the evaluation of both performance models, we use different distance and similarity measures. In Table 5.6, we show the results of the comparison of the generated empirical performance-influence model based on the different sampling heuristics. Additionally, we have executed the distance and similarity measures on the results of the analytical performance model and the measurements and received the results depicted in Table 5.7.

Discussion. Now, that we have presented the results of the 1-dimensional grid, we discuss the obtained results. The analytical performance model has a low error rate of 9.7% in relation to the measurements. Furthermore, the error rates of the empirical performance-influence models to the measurements are between 0.7% and 3.4%.
Considering the results of the distance measures, we observe that they prefer other

sampling heuristics than the error rate. The empirical performance-influence model generated by using the Plackett-Burman design with 49 measurements and 7 levels is preferred by the error rate. The empirical performance-influence model made by using the Plackett-Burman design with 9 measurements and 3 levels is preferred by all similarity and distance measures. By further investigation of the results of the distance measures, we observe that the results are below 1 due to the runtime of the software, which is always below 1 on the 1-dimensional grid. Furthermore, the Minkowski distance has lower results than the Euclidean distance and this, in turn, lower results than the Manhattan distance.

2-Dimensional Grid.

On the 2-dimensional grid, there are additional configuration options compared to the 1-dimensional grid, namely py and ny. However, ny can be replaced by nx due to the constraint that $nx = ny$. As result, the analytical performance model of the 2-dimensional grid from Section 3.1.2 is the following:

$$\begin{aligned} T^{2D} = {} & 4 \cdot \log_2(py) \log_2(px)\alpha + 4 \cdot \log_2(py) \cdot \log_2(nx) \cdot \alpha + 4 \cdot \log_2(nx) \log_2(px) \cdot \alpha \\ & + 4 \cdot \log_2(nx)^2 \cdot \alpha + 4 \cdot \log_2(py) \cdot \alpha + 4 \cdot \log_2(nx) \cdot \alpha + 4 \cdot nx \cdot \log_2(nx) \cdot \beta \\ & + 4 \cdot nx \cdot \log_2(px) \cdot \beta + 4 \cdot nx \cdot \log_2(nx) \cdot \beta + 4 \cdot nx \cdot \log_2(py) \cdot \beta \\ & + 20 \cdot nx^2 \cdot f \end{aligned}$$

Syntactical Comparison. As the 2-dimensional case is more complex, the empirical performance-influence models from SPL Conqueror also become more complex compared to the 1-dimensional case. The achieved results of the syntactic comparison are listed on the first row of Table 5.8. Here, the highest result of the comparison is -12, which is achieved by the Plackett-Burman heuristic with a measurement size of 49 and 7 levels. The corresponding formula of the learned function with the highest result is the following:

$$\begin{aligned} T^{2D}_{PB(49,7)} = {} & -2.279 - 12.6 \cdot \log_{10}(py) - 9.148\text{E-}07 \cdot nx^2 + 7.302 \cdot \log_{10}(px) \\ & + 2.986\text{E-}10 \cdot nx^3 + 3.742 \cdot py - 0.464 \cdot px \end{aligned}$$

In relation to the analytical performance model, the logarithmic influence of py was identified correctly. Apart from py, nx^2 is the only other term that appears in both models. However, the influence of nx^2 has a negative impact on the performance in the empirical performance-influence model, whereas in the analytical performance model it is positive. The logarithmic influence of px also appears in the analytical performance model, but only in an interaction with other configuration options and hence, in different equivalence classes.

In this case, a high syntactic similarity score leads to higher simplicity of the empirical performance-influence model when the analytical performance model is also simple.

Semantical Comparison. As in the 1-dimensional case, we have the problem that the results of the analytical performance model are too low. The results of our original analytical performance model results is presented in Figure 5.3. As before,

Table 5.8: Result of the semantic comparison on 2-dimensional grids.

Measures	FF	PB(9,3)	PB(25,5)	PB(125,5)	PB(49,7)	CC	R(50)	R(125)
SC	−18.9	−19	−12.9	−26	−12	−22	−17.1(±25.9%)	−21(±22.2%)
ER	55.6%	174.7%	103.2%	61.6%	52.5%	126.7%	102.7%(±53%)	111.9%(±39.4%)
CS	91.4%	91.3%	93.6%	89.9%	93.8%	81.9%	91.7%(±1.5%)	92.3%(±0.9%)
JS	66%	63.3%	69.1%	64.1%	69.4%	52.1%	66%(±2.8%)	67.1%(±1.4%)
MD	1.658	1.828	1.468	1.771	1.436	2.540	1.648(±6.5%)	1.584(±3.6%)
ED	5.491	5.858	4.594	6.163	4.277	10.786	5.396(±12.2%)	5.159(±6.2%)
MiD	22.843	25.747	18.644	26.903	19.530	61.251	22.677(±19%)	21.525(±8.5%)
CD	5.353	7.587	5.684	5.784	5.655	12.169	5.765(±12.1%)	5.638(±5%)

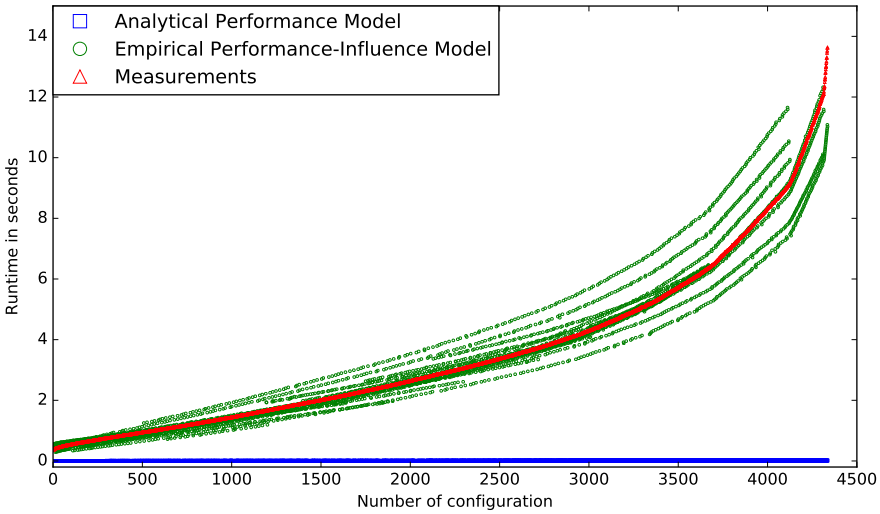

Figure 5.3: The original results of the analytical performance models in relation to the best empirical performance-influence model and the measurements.

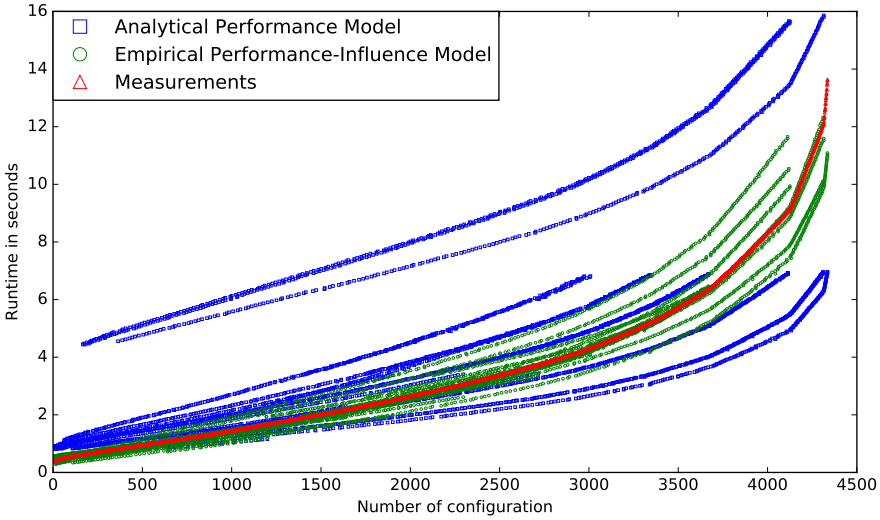

Figure 5.4: The results of the analytical performance models after being multiplied by the factor 500.

we have multiplied the results by factor 500 and have received a better result, as shown in Figure 5.4. In both models, we use the empirical performance-influence

Table 5.9: The results of the different measures when comparing the analytical performance model results of the 2-dimensional grid with the corresponding measurements.

Measures	Results
ER	63%
CS	87.9%
JS	62.5%
MD	1.869
ED	7.152
MiD	34.768
CD	6.702

model with the lowest error rate to the measurements, namely the Plackett-Burman with a measurements size of 125 and a level value of 5 with 9.5%. The error rate of this empirical performance-influence model in relation to the measurements was 9.5%, whereas the worst was 85.3% by Plackett-Burman considering 9 measurements and 3 levels.

Although both performance models depicted in Figure 5.4 show similar behavior, the runtime predictions of both models largely deviate from each other. This behavior is also expressed by applying the distance and similarity measures to the results, as we show in Table 5.8. In the table, the error rate on up to 174.7% is much higher than in the 1-dimensional grid, and accordingly, the Cosine and Jaccard similarity are much lower. The value of the distance metrics are higher than in the 1-dimensional grid, but this may also be a result of the higher runtime of SMG2000.
In Table 5.9, we show the outcome of the comparison between the analytical performance model from the measurements of the 2-dimensional grid. As previously by comparing both types of performance models, we obtain the result, that the analytical performance model has high deviations.

Discussion. Regarding the error rate to the measurements, the results of the empirical performance-influence models range between 9.5% and 85.3%. We observe that the error rate of the empirical performance-influence models is increasing as also the error rate of the analytical performance model to the measurements does. When comparing the best results of the error rate, the similarity measures, the Manhattan and Euclidean distance, then the empirical performance-influence models of the Plackett-Burman with 49 measurements and 7 levels is preferred. Additionally, this empirical performance-influence model has the best result in the syntactic comparison. The only exceptions are the Minkowski and the Chebyshev distance. The former prefers the result of the Plackett-Burman design with 25 measurements and 5 levels, whereas the latter prefers the empirical performance-influence model of the Full Factorial design.
Apart from that, the result of the distance similarity measures do not correspond to the error rate as can be seen on the Full Factorial design and the Plackett-Burman design with 25 measurements and 5 levels. Although the error rate is higher, the similarity of the empirical performance-influence models is not smaller, but higher. Except for the Chebyshev distance, also the distance measures obtain lower distances. This behavior indicates that the error rate, the similarity and the distance

measures behave completely differently from each other.
In contrast to the results on the 1-dimensional grid, the results of the distance measures are above 1. We observe, that now the Minkowski distance has a greater value than the Euclidean distance and this, in turn, a greater value as the Manhattan distance, which is the opposite behavior of the 1-dimensional grid.

3-Dimensional Grid.

The analytical performance model of the 3-dimensional case is the most complex one for the SMG2000 system, as it does allow the configuration options px, py, pz for the processor topology and nx, ny and nz for the grid per processor. Although nx, ny and nz are configurable options, we have to use the same value for all of it, according to Section 3.1.2. To this end, we only use nx as configuration option for the grid size. As a reminder, the formula of the 3-dimensional grid presented in Section 3.1.2 is as follows:

$$\begin{aligned}
T^{3D} = {} & 4 \cdot \log_2(pz) \cdot \alpha + 8 \cdot \log_2(nx) \cdot \alpha + 8 \cdot \log_2(pz) \cdot \log_2(py) \cdot \log_2(px) \cdot \alpha \\
& + 8 \cdot \log_2(pz) \cdot \log_2(py) \cdot \log_2(nx) \cdot \alpha + 8 \cdot \log_2(pz) \cdot \log_2(nx) \cdot \log_2(px) \cdot \alpha \\
& + 8 \cdot \log_2(pz) \cdot \log_2(nx)^2 \cdot \alpha + 8 \log_2(nx) \cdot \log_2(py) \cdot \log_2(px) \cdot \alpha \\
& + 8 \cdot \log_2(nx)^2 \cdot \log_2(py) \cdot \alpha + 8 \cdot \log_2(nx)^3 \cdot \alpha \\
& + 8 \cdot \log_2(pz) \cdot \log_2(py) \cdot \alpha + 8 \cdot \log_2(pz) \cdot \log_2(nx) \cdot \alpha \\
& + 8 \cdot \log_2(nx) \cdot \log_2(py) \cdot \alpha + 8 \cdot \log_2(nx)^2 \cdot \alpha \\
& + 4 \cdot nx^2 \cdot \log_2(pz) \cdot \beta + 4 \cdot nx^2 \cdot \log_2(nx) \cdot \beta + 8 \cdot nx^2 \cdot \log_2(px) \\
& + 8 \cdot nx^2 \cdot \log_2(nx) + 8 \cdot nx^2 \cdot \log_2(py) + 8 \cdot nx^2 \cdot \log_2(nx) \\
& + 48 \cdot nx^3 \cdot f
\end{aligned}$$

Syntactical Comparison. Since we have more configuration options than in the other cases, each performance model has multiple different equivalence classes when performing the syntactical comparison. Thus, the empirical performance-influence models may differ more from the analytical performance model than in the previous cases, as we show in Table 5.10. The best score is -28 and is achieved by performance model using the Full Factorial design, whereas the worst score was -40.9 by the Plackett-Burman design with 9 measurements and 3 levels. The formula of the sampling heuristic with the best score is as follows:

$$\begin{aligned}
T^{3D}_{FF} = {} & 5.411 - 0.968 \cdot \log_{10}(py) - 0.0002 \cdot nx^2 + 2.943\text{E-}06 \cdot nx^3 \\
& + 4.985\text{E-}06 \cdot nx^2 \cdot \log_{10}(py) + 5.662\text{E-}06 \cdot nx^3 \cdot \log_{10}(pz) \\
& + 5.006\text{E-}06 \cdot nx^3 \cdot \log_{10}(px) - 1.715 \cdot \log_{10}(pz) \\
& - 2.343 \cdot \log_{10}(nx) + 3.73\text{E-}06 \cdot nx^3 \cdot \log_{10}(pz) \cdot \log_{10}(px) \\
& + 3.671\text{E-}06 \cdot nx^3 \cdot \log_{10}(py) \cdot \log_{10}(pz) - 0.719 \cdot \log_{10}(px)
\end{aligned}$$

By further investigating this formula, the complexity reminds of the formula in the 2-dimensional grid. In fact, the formula above considers a high number of configuration option interactions, but this does not surprise in relation to the complexity of the analytical performance model, as this also has a higher complexity than the

previous analytical performance models. In comparing both models, we observe that increasing complexity of the empirical performance-influence models means a higher probability of predicting the influence of interactions which do not appear in the analytical performance model. Although the terms $\log_{10}(px)$, $\log_{10}(py)$ and $\log_{10}(pz)$ appear in the empirical performance-influence model, they do not appear in one combination as in the analytical performance model. Additionally, the term nx^2 appears in the predicted model of the Full Factorial design but not in the analytical performance model, which leads to further negative scores. Therefore, different combinations of nx^2 with the other options px, py, pz and nx occur, whereas in the empirical performance model only $nx^2 \cdot log(py)$ is present.

Table 5.10: Best and worst results of the syntactical and semantical comparison strategy in the 3-dimensional grid.

Measures	FF	PB(9,3)	PB(25,5)	PB(125,5)	PB(49,7)	CC	R(50)	R(125)
SC	−28	−40.9	−35.9	−30.9	−34	−34	−35.3(±5.3%)	−36.5(±9.7%)
ER	176.1%	159.5%	208.5%	161.6%	280.8%	107.9%	236.1%(±20%)	319.3%(±98.9%)
CS	79.7%	70.1%	74.8%	80.3%	77.4%	80%	78.4%(±1.5%)	79.8%(±1.7%)
JS	51%	37.6%	47.2%	52.07%	48.6%	52.04%	49.4%(±1.9%)	50.4%(±2.7%)
MD	18.6	28.5	20.8	18.1	20.4	18.9	19.2(±3.3%)	19.3(±4.3%)
ED	981	1 441	1 181	952	1 069	958	1 042(±4.2%)	973(±5.9%)
MiD	68 977	98 011	97 632	68 658	73 223	63 193	78 613(±7.2%)	66 407(±8.1%)
CD	122.88	127.1	134.6	137.1	103.9	122.4	129.8(±7.9%)	117.8(±5.2%)

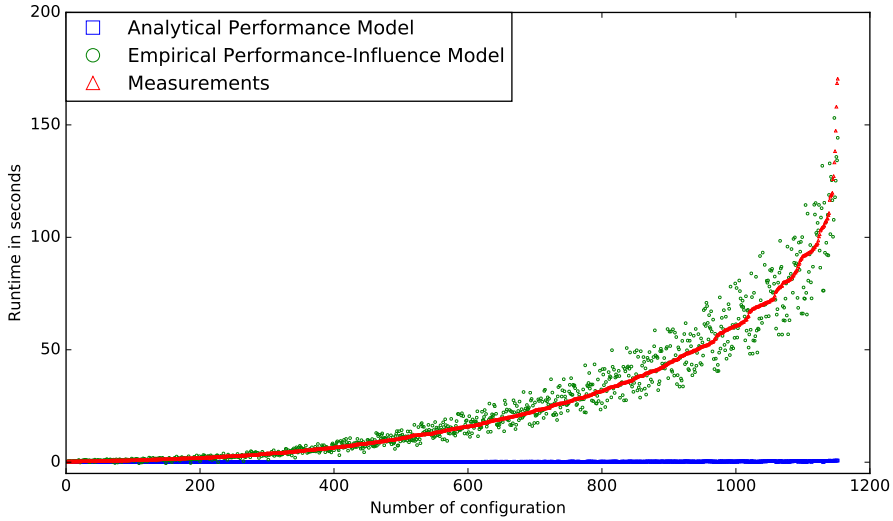

Figure 5.5: Results of the analytical performance model, the measurements and the empirical performance-influence model of the Random design with sample size of 125 and seed 4.

Semantical Comparison. When performing the semantical comparison, we have again recognized that the values of the analytical performance model are far too low. We present the original results in Figure 5.5 and the results after multiplying the α, β and f values with 500 in Figure 5.6. In these figures, the empirical performance-influence model using the Full Factorial design with an error rate of 17.6% in relation to the measurement results is used.

As the deviation of the predictions of the analytical performance model on the 2-dimensional grid was already high, it is even higher in this case, as shown in

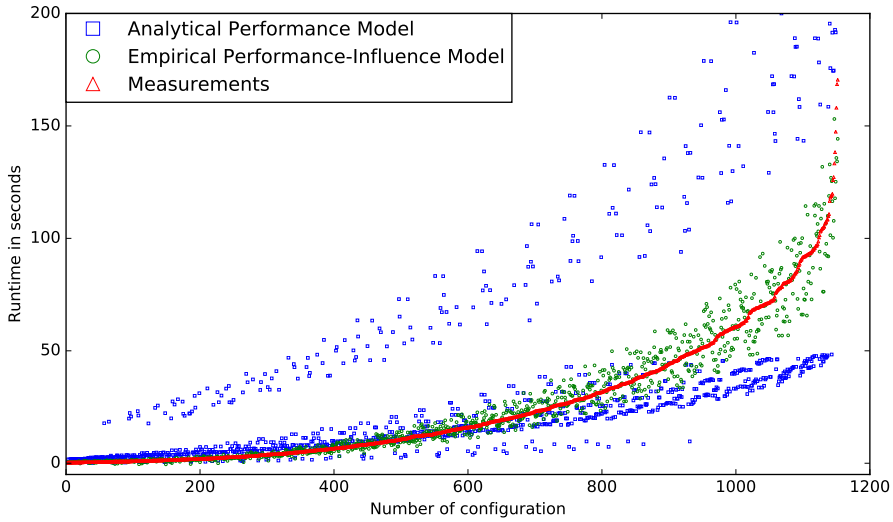

Figure 5.6: Improved results by multiplying the benchmarked values by 500, as before.

Table 5.11: The results of the different measures when comparing the analytical performance model results with the measurements in the case of 3-dimensional grids.

Measures	Results
ER	167%
CS	79.1%
JS	50.4%
MD	19.8
ED	1059
MiD	72948
CD	107.2

Table 5.11. Additionally, the same behavior applies to the comparison between the analytical performance model and the empirical performance-influence model, which we show in Table 5.10. In this case, the empirical performance-influence created with the Central Composite design outperforms the other empirical performance-influence models regarding the error rate.

Discussion. In investigating on the results depicted in Figure 5.6, the results of the analytical performance model are more widespread than on 1-dimensional and 2-dimensional grids. This deviation from the measurement results does explain the high error rate of 167%. Moreover, the error rate of empirical performance-influence models ranges between 17.6% and 420.9%. The worst result was achieved by the model created using the Plackett-Burman design with 9 measurements and 3 levels. Accordingly high are the error rates to the results of the empirical performance-influence models of the different sampling heuristics, which begin at 107.9%. Except for the Minkowski and Chebyshev distance, the distance and similarity measures prefer the empirical performance-influence model created using the Plackett-Burman with 125 measurements and 5 levels. On 3-dimensional grids, the empirical performance-influence model created using the Central Composite design is preferred

by the error rate and the Minkowski distance. The empirical performance-influence model generated using the Plackett-Burman design with 49 measurements and 7 levels is preferred by the Chebyshev distance.

5.4.2 BoomerAMG

We compare the analytical performance model and the empirical performance-influence models of BoomerAMG in this section. As Gahvari et al. [GBS+11] provide multiple analytical performance models by introducing distance, bandwidth and multicore penalty, we consider only the baseline model. To compare both performance models with our program, we have to rewrite the analytical performance model as follows:

$$T = Solve_0 + \sum_{i=1}^{G-1} Solve_i + Solve_{coarsest}$$

$$Solve_0 = 6 \cdot \frac{C_0}{P} \cdot s_0 \cdot t_0 + 3 \cdot p_0 \cdot \alpha + 3 \cdot n_0 \cdot \beta + 2 \cdot \frac{C_1}{P} \cdot \widehat{s}_0 \cdot t_0 + \widehat{p}_0 \cdot \alpha + \widehat{n}_0 \cdot \beta$$

$$\begin{aligned} Solve_{coarsest} = 6 \cdot \frac{C_i}{P} \cdot s_i \cdot t_i + 3 \cdot p_i \cdot \alpha + 3 \cdot n_i \cdot \beta + 2 \cdot \frac{C_{i-1}}{P} \cdot \widehat{s}_{i-1} \cdot t_i + \widehat{p}_{i-1} \cdot \alpha \\ + \widehat{n}_{i-1} \cdot \beta \end{aligned}$$

$$\begin{aligned} Solve_i = 6 \cdot \frac{C_i}{P} \cdot s_i \cdot t_i + 3 \cdot p_i \cdot \alpha + 3 \cdot n_i \cdot \beta + 2 \cdot \frac{C_{i+1}}{P} \cdot \widehat{s}_i \cdot t_i + \widehat{p}_i \cdot \alpha + \widehat{n}_i \cdot \beta \\ + 2 \cdot \frac{C_{i-1}}{P} \cdot \widehat{s}_{i-1} \cdot t_i + \widehat{p}_{i-1} \cdot \alpha + \widehat{n}_{i-1} \cdot \beta \end{aligned}$$

In our reformulation, $Solve_0$ is the formula for level 0, $Solve_{coarsest}$ for the coarsest level and $Solve_i$ for every other level. This is necessary, as the in the finest level no interpolation and on the coarsest level no restriction is performed. We have used SPL Conqueror to learn on the different cycle levels and not on the overall runtime. This has the advantage that the empirical performance-influence models use only the information which is specific to their level, and not the overall information. Furthermore, considering the influences of options, which are related to smaller parts of the code increases the accuracy of the performance prediction, as encountered by Grebhahn et al. [GSKA16]. After learning on all cycle levels, we collect the empirical performance-influence models and use it for both comparison strategies.

Syntactical comparison. The score of achieved in the syntactical comparison are lower than in all dimension cases of SMG2000. The best score, -88.9 is achieved by

the Plackett-Burman design with 9 measurements and 3 levels and the performance model is as follows:

$$\begin{aligned} T = & -417.515 + 2.495\text{E-}05 \cdot t_0^2 + 0.002 \cdot C_0 - 0.021 \cdot C_2 + 221.213 \cdot \log_{10}(C_1) \\ & + 0.005 \cdot \widehat{p}_1^2 - 116.244 \cdot \log_{10}(ny) + 0.279 \cdot p_2 - 150.592 \cdot \log_{10}(ny) \\ & - 31.279 \cdot \log_{10}(C_4) + 0.312305542038874 \cdot p_3 - 157.801 \cdot \log_{10}(ny) \\ & + 0.007 \cdot nx^2 + 18.55 \cdot C_5 - 2.473 \cdot n_4 + 0.008 \cdot ip_4^2 + 1.034 \cdot nx - 0.331 \cdot C_5 \\ & + 0.064 \cdot \widehat{p}_5^2 - 1.304 \cdot C_6^2 + 0.063 \cdot \widehat{p}_4 \end{aligned}$$

As the formula above uses variables, which depend on the level, it does not use all of these variables. For instance, the grid size of all levels has an influence on the performance prediction except for the grid size in level 3, C_3, whereas in the analytical performance model the grid size of every single level is relevant in relation to the overall number of processes. In the empirical performance-influence model, the number of processes, P does not appear directly. Since P is the product of px, py and pz and none of them appears in the empirical performance-influence model neither, the number of processors is not relevant in this empirical performance-influence model.

The formula of the empirical performance-influence model using Plackett-Burman design with 125 measurements and 5 levels obtains the worst score. As a reminder, we have made the observation that the sampling heuristic with the worst score in the 3-dimensional grid of SMG2000 was rather simple when the analytical performance model was more complex than the others. However, this does not apply here, as the formula is so complex that it would fill out 1 full page. Again, we observe that the higher the complexity of both performance models is, the lower the score in the syntactical approach.

Semantical comparison. Now, we investigate on the semantical comparison of BoomerAMG in more detail. Unlike SMG2000, the results of the analytical performance model do not need further adjustments and are close to the measurements. In Figure 5.7, we show the overall results in more than 6000 different configurations in relation with the model learned using the Full Factorial design that has achieved a minimum error rate of 13.2% regarding the measurement results.

In Table 5.12, we show the results of the different distance and similarity measures. The values of the distance measures result are below 1, since the measured runtime is always below 1. However, the similarity and error results are better than on the 3-dimensional grid of SMG2000, because the analytical performance model is more accurate, as we show in Table 5.13.

Discussion. Although the analytical performance model of BoomerAMG does not need further scaling as in SMG2000, the error rate to the measurement results is higher than on the 1-dimensional grid of SMG2000. The error rates of the empirical performance-influence models ranges between 13.2% and 31.3%, which is higher than in the 1-dimensional grid of SMG2000 but lower than in the 2-dimensional grid of SMG2000.
The preferred empirical performance-influence model of all measures except for the Cosine similarity, is the one generated using the Full Factorial design. In contrast to

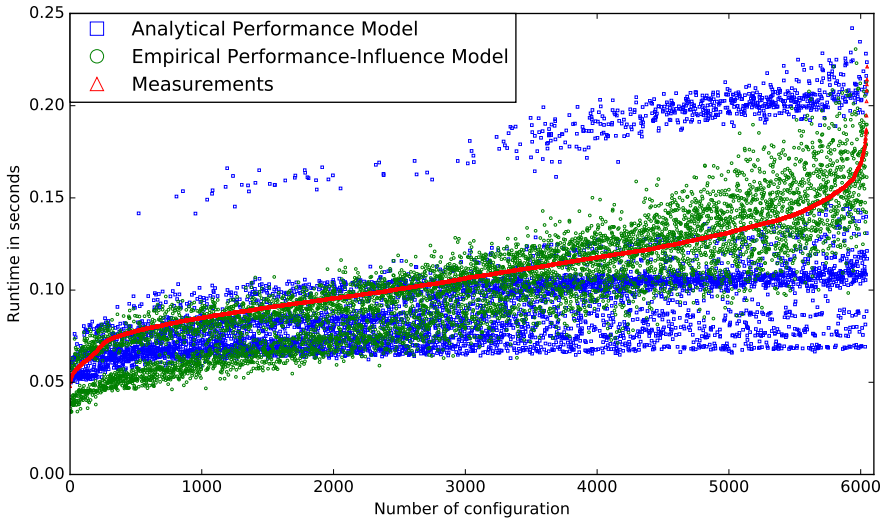

Figure 5.7: Results of the baseline model of BoomerAMG in relation to the Plackett-Burman sampling heuristic and the measurements.

Table 5.12: Results of the syntactical and semantical comparison strategies in BoomerAMG by comparing the results of the analytical performance models and the empirical performance-influence models.

Measures	FF	PB(9,3)	PB(25,5)	PB(125,5)	PB(49,7)	CC	R(50)	R(125)
SC	−156	−88.9	−133.9	−196.9	−181.9	−105	−182(±11.1%)	−164.1(±7.5%)
ER	17.8%	19.8%	22.5%	20.4%	24.7%	46.6%	28.4%(±25.3%)	21.2%(±16%)
CS	96.6%	96.9%	92.4%	95.8%	94.9%	96.9%	94.2%(±1.8%)	96.2%(±0.6%)
JS	82%	78.8%	76.2%	81%	80.2%	72.5%	78.2%(±3.6%)	81.1%(±2.1%)
MD	0.019	0.025	0.027	0.02	0.021	0.035	0.024(±16.2%)	0.02(±12.4%)
ED	0.0007	0.001	0.0019	0.0009	0.001	0.0017	0.0013(±31.3%)	0.0008(±26.2%)
MiD	0.00004	0.00005	0.0002	0.00005	0.0001	0.0001	0.0001(±67%)	0.00005(±44.1%)
CD	0.09	0.15	0.70	0.13	0.31	0.11	0.36(±49.6%)	0.19(±36.1%)

that, the empirical performance-influence model generated using the Central Composite design and Plackett-Burman design with 9 measurements and 3 levels are chosen by the Cosine similarity.
Since the runtime execution values are all below 1, the results of the distance measures are also below 1. We remark that the results of the Minkowski distance are lower than the ones from Euclidean distance and this, in turn lower than the results of the Manhatten distance.
Regarding the analytical performance model, Gahvari et al. [GBS+11] offer further refinings of the baseline model by adding penalties to the latency (α) and the bandwidth (β). However, this is not practical in our case, since this would increase the prediction results further and would lead to a higher error rate regarding the analytical performance results in the scale of 0.15 to 0.25.

Table 5.13: The results of the different measures when comparing the analytical performance model results from the baseline model for BoomerAMG with the measurements.

Measures	Results
ER	22.4%
CS	95.3%
JS	77.6%
MD	0.025
ED	0.0011
MiD	0.00006
CD	0.113

5.5 Discussion

Since we have presented the achieved results by the comparison in the last sections, we summarize the results and answer the questions from Chapter 1, which are answered in the following.

RQ1: Is it possible to identify the influences proposed by analytical performance models?

On the base of the previous evaluation, we are able to answer this question. In considering the syntactical similarity between analytical performance models and empirical performance-influence models, we have obtained similar, but no equal empirical performance-influence models. This may be caused due to the fact that the domain experts have not considered certain influence factors or declared them as unrelevant, although they are relevant. Since the syntactical score was always negative, we have not managed to learn performance models with the same terms and also not with the same influences. Therefore, we answer to this question with ***no*** with respect to the tool SPL Conqueror and the used sampling heuristics.

RQ2: Does the sampling strategy have a high influence on the similarity of empirical performance-influence models to analytical performance models?

For answering this question, we relate to both comparison strategies, which calculate the similarity between two performance models based on the syntactical and semantical information of the performance models. In this work, we have investigated the Full Factorial, Random, Plackett-Burman and the Central Composite designs as sampling heuristics for the generation of empirical performance-influence models, which we consider next.

Full Factorial Design. Since the Full Factorial design takes the whole configuration space as learning set, this sampling heuristic has the lowest error rates in relation to the measurements. Moreover, the error rate from the analytical performance model to the measurements is always higher than the error rate from the empirical performance-influence model generated using the Full Factorial design. To

this end, the error rate in relation to the analytical performance models of SMG2000 is only middle-rate. Furthermore, the error rate to the analytical performance model is always very similar to the error rate of the analytical performance model with the measurement. In relation to the syntactical comparison, this design has received the best scores in SMG2000. The only exception is the 2-dimensional case of SMG2000, where this design has received moderate results. Although this design has achieved good scores in the both comparison strategies of SMG2000, the requirement of measuring all configurations in the configuration space is oftenly too difficult to satisfy.

Random Design. Since the Random design is based on a seed, which may be selected randomly, we have presented the mean value of the resulting empirical performance-influence models of the Random design and the according standard deviation. In our results, the standard deviation of the Random design lasted between 2-98.9% with respect to the error rate to the analytical performance model. In fact it has turned out that the Random design oftenly contains the best but in the same time the worst results in relation to Plackett-Burman and Central Composite. Furthermore, the sample size of the Random design has lead to different results in both case studies, where none clearly outperforms the other. To sum it up, the Random design is no good choice if one is interested in finding a syntactically or semantically similar performance model.

Plackett-Burman Design. In our syntactical comparison to the analytical performance models, the Plackett-Burman design shows the best results in relation to the Central Composite and Random design. Only in the 1-dimensional case of SMG2000, Plackett-Burman was slightly worse than Central Composite. Furthermore, this design has the lowest error rate in the semantical comparison to the analytical performance models in BoomerAMG and the 1-dimensional grid of SMG2000, where the error rate of the analytical performance model is below 10%. Comparing the presented Plackett-Burman configuration with each other, the designs with 9 measurements and 3 levels and with 49 measurements and 7 levels alternatingly achieved the best results.

Central Composite Design. The results of the Central Composite design in the syntactical comparison were never the worst in our case studies. Furthermore, the score of the syntactical comparison outperformed the Random and Plackett-Burman sampling heuristics only in the 1-dimensional. In relation to the semantic comparison, this sampling heuristic has achieved the best results when the error of the analytical performance model to the measurements was the highest.

Based on the observations on the different designs, the selection of the design has in fact a high influence on the similarity of the result empirical performance-influence model and the analytical performance model. Hence, the answer to this question is ***yes***. Unfortunately, we have not found a sampling heuristic that always performs best in this task.

RQ3: Is it beneficial to compare analytical and empirical performance models using only syntactic information?

By considering the results from both case studies, the score of the syntactical comparison was always negative. This is because the cases, where either the term or the whole equivalence class was not present in one of both models mostly applied in our work, whereas the case that the influence of two terms is exactly the same has never applied. According to our results, a score that is greater than -10 in relation to a rather simple analytical performance model resulted in an empirical performance-influence model, which was similarly simple. Although we have found no empirical performance-influence model having a positive score, the syntactical comparison provides information about how many relevant and how many unrelevant terms were found. To this end, comparing analytical and empirical performance models using only syntactic information is beneficial, so the answer is ***yes***.

RQ4: Is it beneficial to compare analytical and empirical performance models using distance and similarity measures?

To find an answer to this question, we have created the semantical comparison in such a way that it is based on different distance and similarity measures. The findings regarding the semantical comparison are explained next.

In the semantical comparison strategy that we describe in Section 4.2, we have evaluated the results of the analytical performance models and the empirical performance-influence models based on different distance and similarity measures and the error rate.
Firstly, the error rate has shown that it is an indicator of how much different the results of two performance models are. Although this relation is expressed in percent, the results of the error rate sometimes exceeds 100% and thus, is a less good indicator for the percentual dissimilarity. In contrast to dissimilarity, the Cosine and Jaccard similarity have measured the percentual similarity of two performance models. When examining the results, the results of the Cosine similarity are mostly above 90%. Since this similarity measure considers the angle between the results of different performance models, little deviations of small numbers may have the same relevance as high deviations of higher values. This property can be observed when comparing the results of the analytical performance model with the measurements in the 2-dimensional and 3-dimensional case of SMG2000. The Jaccard similarity shows a better behavior in these cases and mostly results in a lower similarity, when the error rate is high. Moreover, the results of the Jaccard similarity vary more than the ones received by the Cosine similarity and admits a easier comparison between performance models. When considering the distance metrics from the Minkowski family, it turns out that they behave differently for execution times below and over 1. In the case that the execution times are mostly below 1, the value of the Minkowski measure is lower than the one achieved from the Manhattan distance and the Euclidean distance. If the most execution times are above 1, the results from the Minkowski measure are higher than the other distance measures. To overcome this behavior, one could scale the execution times above 1. As some of our results

contained both values less than 1 and greater than 1, the behavior may become unpredictable.
Moreover, the Chebyshev distance shows the greatest distance between all configurations, which may be helpful, but not when comparing two large sets of points, as it would only register the distance of outliers. A similar behavior is shown in Table 5.9 and Table 5.11.
We have found that the Jaccard similarity produces the best results for comparing such performance models, although the other distance and similarity measures could also be used. As we have seen before, the distance and similarity measures behave differently. To this end, one should know the requirements that the distance or similarity measure should satisfy. To sum it up, distance and similarity measures can be used for the comparison of two performance models and hence, the answer to the question is ***yes***.

RQ5: Do the results of the similarity and distance measures correlate with the error rate?

Additionally to the observations for ***RQ4***, we have observed that none of the chosen distance and similarity measures behaves as the error rate. Thus, the answer on this question is ***no***.

Further Findings.

An unexpected finding that we have made in this work is related to the results of the analytical performance model of BoomerAMG. There, the most results are below or near the results obtained by the measurements. However, some predictions of the analytical performance model were too high. After a deeper investigation, we figured out that these results were all related to the measurements consisting of 4 nodes. A reason for the deviation could be that the latency and the bandwidth was better than measured by the benchmarks when performing the software with 4 nodes.
A similar observation can be made in SMG2000, where the results of the analytical performance models can be ordered in lines. These lines stand for the different number of processes.

Another surprising finding was the scale of the analytical performance model by factor 500. The reasons for this behavior are diverse. This behavior could be a result of using another architecture, but also a scale error by the authors from Brown et al. [BFJ00].

5.6 Threats to Validity

Internal Validity. For the minimization of measurement bias during the measurements on Chimaira, we have run each configuration 3 times and calculated the standard deviation. The standard deviation of each configuration was less than 10%. Due to time restrictions, we have executed the measurements in parallel. As this could affect the latency and the bandwidth, we have additionally executed tests

without interferences. However, the results from the parallel execution coincides with the results of our tests.

To rule out errors in the implementation of the analyzation, we have calculated the syntactical comparison as well as the semantical comparison for a set of points by hand. The only deviation in the computation may be caused by using the datatype `double`, which is capable of storing only a fixed amount of decimal places and thus, introduces a minimal inaccuracy. Moreover, we have compared the results of the empirical performance-influence models with the results achieved by SPL Conqueror, which coincided with each other.

Furthermore, we have calculated some configurations by hand and have received the same results in SMG2000 as our implementation, which is an error in the analytical performance model. Multiplying the benchmarked values by the value 500 was achieved by different tests. A reason for this behavior is the simplicity of the analytical performance models for SMG2000. Although it is easy to understand, the analytical performance model has left out important aspects, which were in turn considered by the analytical performance model from BoomerAMG.

External Validity. In this thesis, we were focused on two different configurable multigrid systems. Nevertheless, our comparison strategies are generally applicable on different software systems that provide different configuration parameters for the selection or deselection of different components that mostly have an influence on the performance of the system. This type of comparison can not only be applied to analytical performance models and empirical performance-influence models but also on different performance models of the same kind. For instance, our comparison can be applied to empirical performance-influence models of different sampling heuristics.

6. Related Work

To our best knowledge, analytical performance models and empirical performance-influence models of multigrid systems were not compared in a publication yet. However, the strengths and weaknesses of analytical performance models and empirical performance-influence models, as described in Section 2.1, are widely known. For building better performance models, Didona et al. [DQRT15] combines analytical performance models with empirical performance-influence models to build three different gray-box approaches. In their evaluation on two case-study systems, none of these gray-box approaches clearly outperforms the other ones.

In this thesis, we used the analytical performance models for SMG2000 and BoomerAMG. As mentioned earlier, analytical performance models are not created specifically for multigrid systems, but also for other popular software. One analytical performance model is proposed by Yang et al. [YS11], where the performance of MapReduce is modeled. Based on the analytical performance models, the behavior of MapReduce is analyzed. The result of this analyzation is that configuration options with a great impact on the performance are found and improvements by selecting specific values for numerical options are conducted. Another analytical performance model is proposed by Hong et al. [HK09]. The goal of their work is to create an analytical performance model for GPU architectures to help finding performance bottlenecks of parallel applications that are executed on the GPU. Furthermore, their analytical performance model is evaluated on two different GPU's.

In this work, we use SPL Conqueror for generating empirical performance-influence models on the base of different sampling heuristics. Besides, there are other tools, such as Catwalk that is proposed by Wolf et al. [WBH+14]. In contrast to SPL Conqueror, their main goal is to find *scalability bugs* of parallel software and not the prediction of the influence of multiple configuration options and interactions between those. Besides, there are tools for building empirical performance-influence models in a white-box manner such as Palm [TH14]. Palm is able to create empirical performance-influence models such as the one of SMG2000 presented in Section 3.1.2, where the MPI functions like `MPI_Isend` are taken in consideration. This is done by

inserting annotations in the code as we have done it using TAU and by monitoring different regions of the code.

7. Conclusion

In this section, we summarize the findings of this work in Section 7.1. Afterwards, further enhancements and future work of this thesis are presented in Section 7.2.

7.1 Summary

In this thesis, we have compared analytical performance-influence models from two multigrid case studies with different empirical performance-influence models. The empirical performance-influence models were generated by SPL Conqueror based on a set of configurations defined by different sampling heuristics. This comparison was done on the base of two different comparison strategies, namely the *syntactical* and the *semantical* comparison. Based on this comparison strategies, we have discussed multiple aspects.

Firstly, we have investigated if SPL Conqueror is able to find the same terms as the domain experts in their analytical performance model. Although some empirical performance-influence models have considered almost all terms, we have not found one empirical performance-influence model which was equal to the analytical performance model.

Moreover, we examine the influence of different sampling heuristics with the other sampling heuristics in relation to the results achieved by syntacical comparison strategy. We have found out that the Full Factorial design does not outperform other sampling heuristics when comparing them with analytical performance models. This means that considering all configurations of the configuration space is not necessary to find empirical performance-influence models that are similar to the analytical performance models. One further finding was that the Random design achieved a deviation on up to 98.9% in syntactic comparison. Due to the high deviations of the Random design, the outcome of the Random design is uncertainly. In addition to that, we have encountered that increasing the sampling set in the Random design sampling heuristic does not necessarily improve the empirical performance-influence models generated by SPL Conqueror. Furthermore, the empirical performance-influence models using Plackett-Burman design with 9 measurements and 3 levels as

well as with 49 measurements and 7 levels achieved the best results in the syntactical comparison strategy in relation to the other sampling heuristics when the analytical performance model had a higher accuracy. We have shown that the chosen sampling heuristic has in fact an influence on the result of both comparison strategies, even though no empirical performance-influence model of our sampling heuristics has clearly outperformed the others.

Besides, we observed that the syntactic comparison is a good indicator for the similarity of two performance models, even though the results obtained by this comparison strategy were always negative. Furthermore, we have compared the application of different distance and similarity measures on the results of comparison between the analytical performance models and the empirical performance-influence models. For the Minkowski distance measures, we have made out that they do not behave well when the result set contains execution times below 1. Additionally, we observed that the Chebyshev distance has not provided useful information in the semantic comparison. Furthermore, the Cosine similarity turned out to be less suitable, since the similarity result was largely above 90%, even if the error between the analytical performance model and the empirical performance-influence model was high. Among the chosen distance and similarity measures, the Jaccard similarity turned out to be the best in our study. Finally, we investigated if the results of the distance and similarity measures correspond to the one of the error rate and figured out that they are independent from each other.

7.2 Future Work

This work can be taken as the base on further research on the comparison of performance models. One improvement could be alternating grid size in SMG2000. In the work of Brown et al. [BFJ00], the grid size in the analytical performance model was limited to a cube and resulted in the constraint that $nx = ny = nz$. In the future, we could investigate the impact of the grid size in different dimensions, which would provide further information about the behavior of the respective software and on the generality of the analytical performance models.
Furthermore, the number of pre-smoothing and post-smoothing steps presented in Section 2.4 were fixed in both models. Thus, the influence of smoothing in BoomerAMG and SMG2000 could not be observed.
An additional improvement is related to the processor topology of both case studies. For instance, in the work of Gahvari et al. [GBS+11], where the analytical performance models of BoomerAMG were created, the analytical performance models were partially evaluated on clusters with thousands of cores. Increasing the number of processes could provide additional information according to the communication pattern.
Different multigrid cycles, as mentioned in Section 2.4, depict another property of both multigrid systems that were not investigated further. Only the simplest cycle, the V-cycle was performed in all our measurements. Since the different cycles have an impact on the number of calls of each level as well as on the final convergence rate [TS01], this configuration would be worth further investigation.

Moreover, the variability of distance measures and similarity measures presented

in Section 2.3 can additionally be expanded. In our work, we were focused on unweighted measures. Thus, another improvement would be the use of weighted measures, where configuration option combinations having a great influence at the performance would have a higher weight than configuration option combinations with less impact. In addition to that, we could investigate further distance measures and similarity-measure families. Throughout the evaluation, we were focused on the Minkowski- and the inner-product family. Some different distance- and similarity-measures were summarized in Cha et al. [Cha07].

Another future work is the improvement of the semantical comparison. Since we have applied our distance and similarity measures on point data, we could additionally apply them on interval data. Therefore, a way to express performance models as an interval has to be found.
Besides, the minimum and maximum value of each term according to the configuration space can be computed to weight the syntactical and semantical comparison according to their relevance.

Furthermore, a hybrid comparison strategy is another future work. This hybrid comparison strategy could use the syntactical as well as the semantical information of performance models for the comparison. Therefore, the influence of the equivalence classes on the performance could be computed. According to the percentual influence of an equivalence class, the semantic and syntactic comparison could be weighted.

A. Appendix

A.1 Folder and File Structure on the CD

In this thesis, we have compared analytical performance models with empirical performance-influence models from two multigrid-case studies, namely SMG2000 and BoomerAMG. The empirical performance-influence models were generated using SPL Conqueror and the comparison of the performance models was performed by using our tool PErformance MOdel COmparer (PeMoCo). Both tools, written

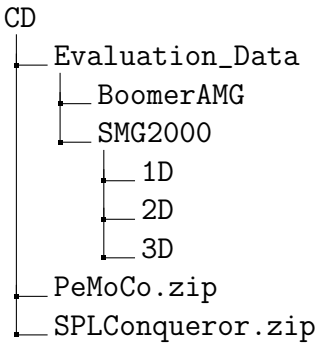

Figure A.1: The folder hierarchy on the CD. In the directories of the folder *Evaluation_Data*, the measurement results and tool output is included.

with C#, are located as zip files on the CD. In the zip folder of both tools, `sln` files are included for opening these projects by an integrated development environment (IDE), such as Visual Studio or MonoDevelop. Furthermore, we have included the measurement data of SMG2000 and BoomerAMG, as well as the output of both tools. We show the folder hierarchy in Figure A.1.

Generally, in each folder of both case studies, we have included the equations of the analytical performance model (`analyticalModel.txt`), the system description (`boomeramg.xml` and `smg2000.xml`), the values obtained by the benchmarks (`depVals.txt`). Furthermore, the output data of SPL Conqueror is written in the files ending with `log`. On these `log`-files, we have applied PeMoCo, whose results are

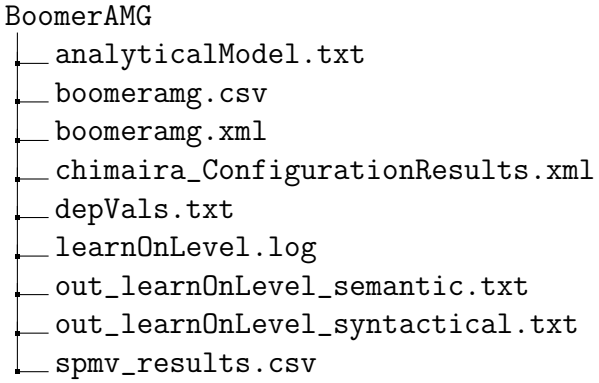

Figure A.2: The files in the BoomerAMG folder.

written in the files ending with `syntactic.txt` and `semantic.txt`. In Figure A.2, we show the files in the folder of BoomerAMG and in Figure A.3 for SMG2000. Note that the file structure for each dimension of SMG2000 is the same. The files in the folder of BoomerAMG differs slightly from the files in each folder of SMG2000, as we have collected more data than in SMG2000. We have stored the overall performance and instrumentation results in `chimaira_ConfigurationResults.xml` and the results achieved by the benchmark SPmV in `spmv_results.csv`.

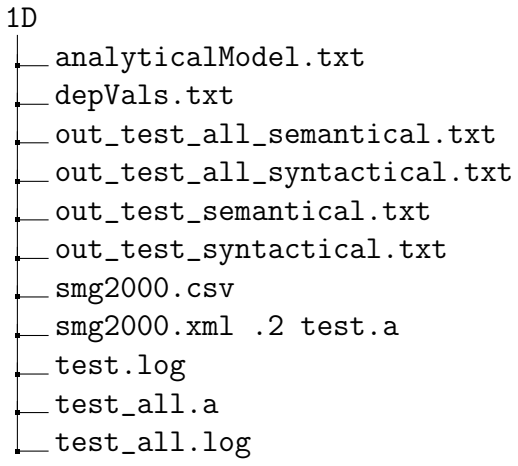

Figure A.3: The files in the SMG2000 folder for each dimension.

Bibliography

[BDGGW05] A. Barchanski, H. De Gersem, E. Gjonaj, and T. Weiland. Impact of the Displacement Current on Low-Frequency Electromagnetic Fields Computed Using High-Resolution Anatomy Models. *Physics in Medicine and Biology*, pages 243–249, 2005. (cited on Page 22)

[BFJ00] P. N. Brown, R. D. Falgout, and J. E. Jones. Semicoarsening Multigrid on Distributed Memory Machines. *SIAM Journal on Scientific Computing*, 21(5):1823–1834, 2000. (cited on Page 19, 20, 34, 36, 49, and 54)

[Bis06] C. M. Bishop. Pattern Recognition. *Machine Learning*, 128, 2006. (cited on Page 5)

[Bot10] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPuter STATistics(COMPSTAT)'2010*, pages 177–186. Springer, 2010. (cited on Page 5)

[BS07] S. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods*, volume 15. Springer Science & Business Media, 2007. (cited on Page 15)

[BTA10] D. Bingol, N. Tekin, and M. Alkan. Brilliant Yellow Dye Adsorption onto Sepiolite Using a Full Factorial Design. *Applied Clay Science*, 50(3):315–321, 2010. (cited on Page 9)

[BW51] J. Box and W. Wilson. Central Composites Design. *Royal Statistical Society*, 1, 1951. (cited on Page 10)

[Car01] B. Carnes. The SMG2000 Benchmark Code, 2001. (cited on Page 2 and 19)

[CFJ98] A. J. Cleary, R. D. Falgout, and J. E. Jones. Coarse-Grid Selection for Parallel Algebraic Multigrid. In *International Symposium on Solving Irregularly Structured Problems in Parallel*, pages 104–115. Springer, 1998. (cited on Page 22)

[Cha07] S. Cha. Comprehensive Survey on Distance/Similarity Measures Between Probability Density Functions. *Mathematical Models and Methods*, 1(2):300–307, 2007. (cited on Page 13 and 55)

[CLB03] C. Chen, H. Liu, and R. C. Beardsley. An Unstructured Grid, Finite-Volume, Three-Dimensional, Primitive Equations Ocean Model: Application to Coastal Ocean and Estuaries. *Journal of Atmospheric and Oceanic Technology*, 20(1):159–186, 2003. (cited on Page 22)

[CS14] G. Chandrashekar and F. Sahin. A Survey on Feature Selection Methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014. (cited on Page 5)

[dABK+95] P. F. de Aguiar, B. Bourguignon, M. S. Khots, D. L. Massart, and R. Phan-Than-Luu. D-Optimal Designs. *Chemometrics and Intelligent Laboratory Systems*, pages 199–210, 1995. (cited on Page 8)

[DB79] D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979. (cited on Page 12)

[DLP03] J. J. Dongarra, P. Luszczek, and A. Petitet. The linpack benchmark: past, present and future. *Concurrency and Computation: Practice and Experience*, 15(9):803–820, 2003. (cited on Page 30)

[DQRT15] D. Didona, F. Quaglia, P. Romano, and E. Torre. Enhancing Performance Prediction Robustness by Combining Analytical Modeling and Machine Learning. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pages 145–156. ACM, 2015. (cited on Page 51)

[DR15] D. Didona and P. Romano. Hybrid Machine Learning/Analytical Models for Performance Prediction: A Tutorial. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pages 341–344. ACM, 2015. (cited on Page 4 and 5)

[dZ96] P. M. de Zeeuw. Development of Semi-Coarsening Techniques. *Applied Numerical Mathematics*, 19(4):443–465, 1996. (cited on Page 19)

[FY02] R. D. Falgout and U. M. Yang. Hypre: A Library of High Performance Preconditioners. In *International Conference on Computational Science*, pages 632–641. Springer, 2002. (cited on Page 19 and 22)

[Gah06] H. Gahvari. Benchmarking Sparse Matrix-Vector Multiply. Master's thesis, Citeseer, 2006. (cited on Page 30)

[GBS+11] H. Gahvari, A. H. Baker, M. Schulz, U. M. Yang, K. E. Jordan, and W. Gropp. Modeling the Performance of an Algebraic Multigrid Cycle on HPC Platforms. In *Proceedings of the International Conference on Supercomputing*, pages 172–181. ACM, 2011. (cited on Page 22, 23, 32, 43, 45, and 54)

[GSKA16] A. Grebhahn, N. Siegmund, H. Köstler, and S. Apel. Performance Prediction of Multigrid-Solver Configurations. In *Software for Exascale Computing – SPPEXA 2013–2015*, volume 113 of *Lecture Notes*

in Computational Science and Engineering, pages 69–88. Springer, September 2016. (cited on Page 43)

[GV02] P. Giannopoulos and R. C. Veltkamp. A Pseudo-Metric for Weighted Point Sets. In *European Conference on Computer Vision*, pages 715–730. Springer, 2002. (cited on Page 12)

[HHH+89] L. Hamers, Y. Hemeryck, G. Herweyers, M. Janssen, H. Keters, R. Rousseau, and A. Vanhoutte. Similarity Measures in Scientometric Research: the Jaccard Index Versus Salton's Cosine Formula. *Information Processing & Management*, 25(3):315–318, 1989. (cited on Page 13)

[HK09] S. Hong and H. Kim. An Analytical Model for a GPU Architecture with Memory-Level and Thread-Level Parallelism Awareness. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 152–163. ACM, 2009. (cited on Page 4 and 51)

[HLP52] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1952. (cited on Page 14)

[KH14] G. Kourakos and T. Harter. Parallel Simulation of Groundwater Non-Point Source Pollution Using Algebraic Multigrid Preconditioners. *Computational Geosciences*, 18(5):851–867, 2014. (cited on Page 22)

[Kry14] M. Kryszkiewicz. The Cosine Similarity in Terms of the Euclidean Distance. *Encyclopedia of Business Analytics and Optimization*, pages 2498–2508, 2014. (cited on Page 13)

[LeV97] R. J. LeVeque. Wave Propagation Algorithms for Multidimensional Hyperbolic Systems. *Journal of Computational Physics*, 131(2):327–353, 1997. (cited on Page 14)

[MDHS09] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney. Producing Wrong Data Without Doing Anything Obviously Wrong! *ACM Sigplan Notices*, 44(3):265–276, 2009. (cited on Page 4)

[MSB91] H. Mühlenbein, M. Schomisch, and J. Born. The Parallel Genetic Algorithm as Function Optimizer. *Parallel Computing*, 17(6-7):619–632, 1991. (cited on Page 5)

[NB10] H. V. Nguyen and L. Bai. Cosine Similarity Metric Learning for Face Verification. In *Asian Conference on Computer Vision*, pages 709–720. Springer, 2010. (cited on Page 13)

[NIS16] NIST/SEMATECH. E-Handbook of Statistical Methods. *http://www.itl.nist.gov/div898/handbook/*, (09/27/2016). (cited on Page 11)

[PB46] R. L. Plackett and J. P. Burman. The Design of Optimum Multifactorial Experiments. *Biometrika*, 33(4):305–325, 1946. (cited on Page 9)

[Puk06] F. Pukelsheim. *Optimal Design of Experiments*. Classics in Applied Mathematics, 2006. (cited on Page 8)

[RS87] J. W. Ruge and K. Stüben. Algebraic Multigrid. *Multigrid Methods*, 3(13):73–130, 1987. (cited on Page 22)

[SGAK15] N. Siegmund, A. Grebhahn, S. Apel, and C. Kästner. Performance-Influence Models for Highly Configurable Systems. In *Proceedings of the Joint Meeting on Foundations of Software Engineering*, pages 284–294. ACM, 2015. (cited on Page 6)

[SM06] S. S. Shende and A. D. Malony. The TAU Parallel Performance System. *International Journal of High Performance Computing Applications*, pages 287–311, 2006. (cited on Page 23 and 30)

[SRK+12] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake. SPL Conqueror: Toward Optimization of Non-Functional Properties in Software Product Lines. *Software Quality Journal*, 20(3-4):487–517, 2012. (cited on Page 5)

[Tay13] Y. C. Tay. Analytical Performance Modeling for Computer Systems. *Synthesis Lectures on Computer Science*, 4(3):1–141, 2013. (cited on Page 4)

[Tea02] MVAPICH Team. MVAPICH 1.2 User and Tuning Guide. Technical report, The Ohio State University, 2002. (cited on Page 30)

[TH14] N. R. Tallent and A. Hoisie. Palm: Easing the Burden of Analytical Performance Modeling. In *Proceedings of the ACM International Conference on Supercomputing*, pages 221–230. ACM, 2014. (cited on Page 51)

[TS01] U. Trottenberg and A. Schuller. *Multigrid*. Academic Press, Inc., 2001. (cited on Page 1, 14, 17, and 54)

[WBH+14] F. Wolf, C. Bischof, T. Hoefler, B. Mohr, G. Wittum, A. Calotoiu, C. Iwainsky, A. Strube, and A. Vogel. Catwalk: A Quick Development Path for Performance Models. In *European Conference on Parallel Processing*, pages 589–600. Springer, 2014. (cited on Page 5 and 51)

[XX11] Z. Xu and M. Xia. Distance and Similarity Measures for Hesitant Fuzzy Sets. *Information Sciences*, 181(11):2128–2138, 2011. (cited on Page 12)

[Yan02] U. M. Yang. BoomerAMG: A Parallel Algebraic Multigrid Solver and Preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002. (cited on Page 2 and 22)

[YS11] X. Yang and J. Sun. An Analytical Performance Model of Mapreduce. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 306–310. IEEE, 2011. (cited on Page 4 and 51)

Eidesstattliche Erklärung:

Hiermit versichere ich an Eides statt, dass ich diese Masterarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie dass ich die Masterarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Christian Kaltenecker

Passau, den 29. November 2016