Seminar/Proseminar

Modellbasierte Entwicklung

Sommersemester 2024

Organisation

Bachelor: Proseminar **Master:** Seminar

- **Wissenschaftliche Präsentation** zum gewählten Thema im Rahmen einer zentralen Veranstaltung am Ende des Semesters.
  - Proseminar: 20 Minuten Vortrag + 10 Minuten Diskussion
  - Seminar: 30 Minuten Vortrag + 15 Minuten Diskussion
- **Schriftliche Ausarbeitung** des gewählten Themas.
  - Proseminar: 6 Seiten zweispaltig / 9 Seiten einspaltig
  - Seminar: 10 Seiten zweispaltig / 15 Seiten einspaltig
  - ACM Style LaTeX-Vorlage muss verwendet werden

Meilensteine

1. Besprechung der Literatur und erster Entwurf der Gliederung der Ausarbeitung (ca. 4 Wochen nach der Einführungsveranstaltung)
2. Abgabe eines ersten Zwischenstandes der Ausarbeitung und Anfertigen von Reviews für die Ausarbeitungen anderer Teilnehmer (ca. 4 Wochen vor dem Vortragstermin)
   - Proseminar: 1 Review
   - Seminar: 2 Reviews
3. Abgabe eines ersten Präsentationsentwurfs und Probevortrag (ca. 2 Wochen vor dem Vortragstermin)

Vortragstermin: voraussichtlich Ende Juli (folgt per E-Mail).

Abgabe der Vortragsfolien: spätestens am Tag des Vortrags

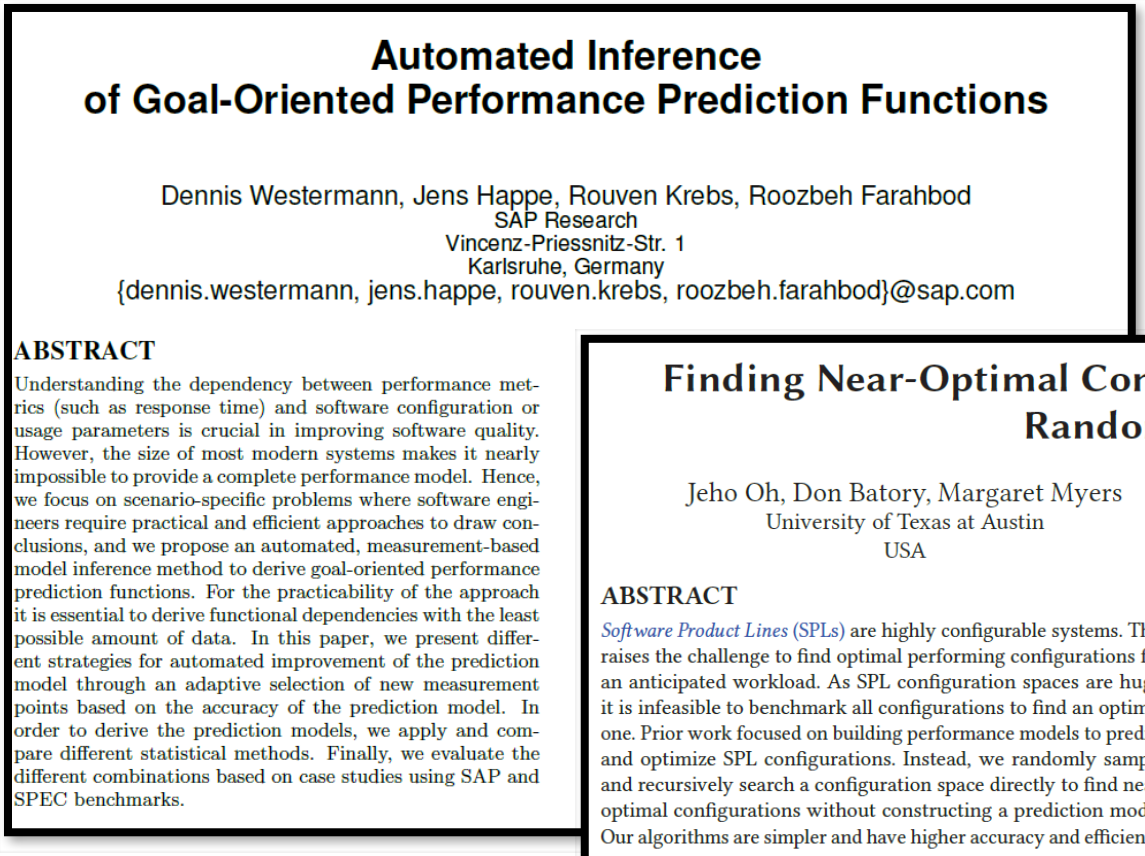
Abgabe der schriftlichen Ausarbeitung: spätestens am 31.08.2024

Informationsquellen

1. Unisono-Rundmail:
   - Allgemeine Ankündigungen
2. MBE-Webseite*:
   - Organisatorische Informationen (inkl. diese Folien)
   - Anleitung für die ACM Style LaTeX-Vorlage
   - PowerPoint-Folienmaster für die Abschlusspräsentation
3. Ihr Themenbetreuer:
   - Besprechung der Meilensteine
   - Individuelle Fragen

* https://mbe.informatik.uni-siegen.de/lehre/sommersemester/modellbasierte-entwicklung/

Thema 1 – Finding Near-Optimal Configurations

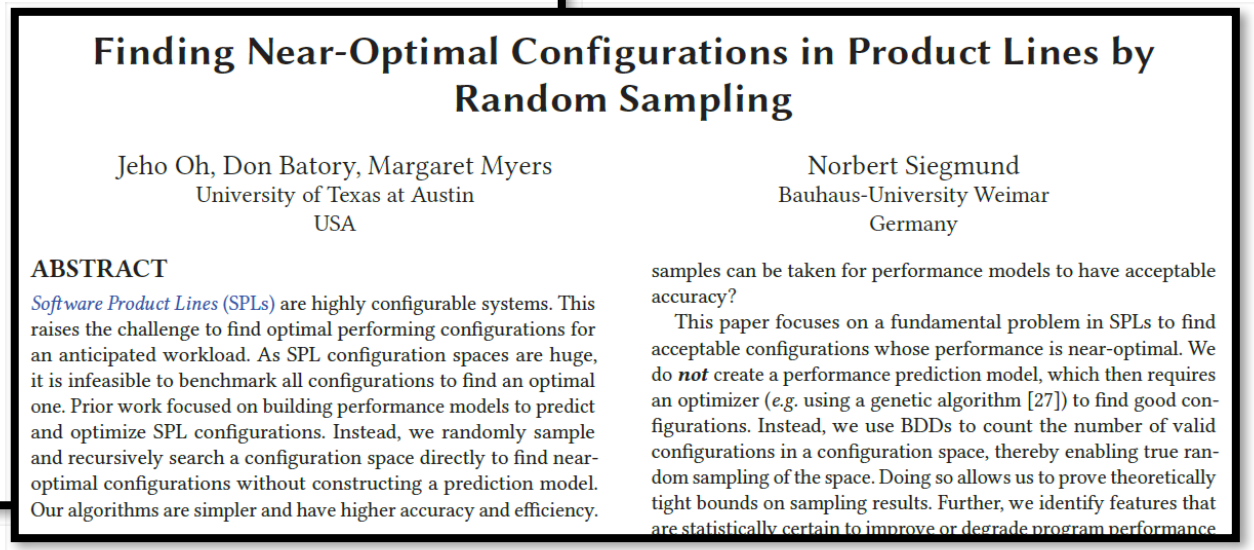


Automated Inference of Goal-Oriented Performance Prediction Functions

Dennis Westermann, Jens Happe, Rouven Krebs, Roozbeh Farahbod
SAP Research
Vincenz-Priessnitz-Str. 1
Karlsruhe, Germany
{dennis.westermann, jens.happe, rouven.krebs, roozbeh.farahbod}@sap.com

ABSTRACT

Understanding the dependency between performance metrics (such as response time) and software configuration or usage parameters is crucial in improving software quality. However, the size of most modern systems makes it nearly impossible to provide a complete performance model. Hence, we focus on scenario-specific problems where software engineers require practical and efficient approaches to draw conclusions, and we propose an automated, measurement-based model inference method to derive goal-oriented performance prediction functions. For the practicability of the approach it is essential to derive functional dependencies with the least possible amount of data. In this paper, we present different strategies for automated improvement of the prediction model through an adaptive selection of new measurement points based on the accuracy of the prediction model. In order to derive the prediction models, we apply and compare different statistical methods. Finally, we evaluate the different combinations based on case studies using SAP and SPEC benchmarks.



Finding Near-Optimal Configurations in Product Lines by Random Sampling

Jeho Oh, Don Batory, Margaret Myers
University of Texas at Austin
USA

Norbert Siegmund
Bauhaus-University Weimar
Germany

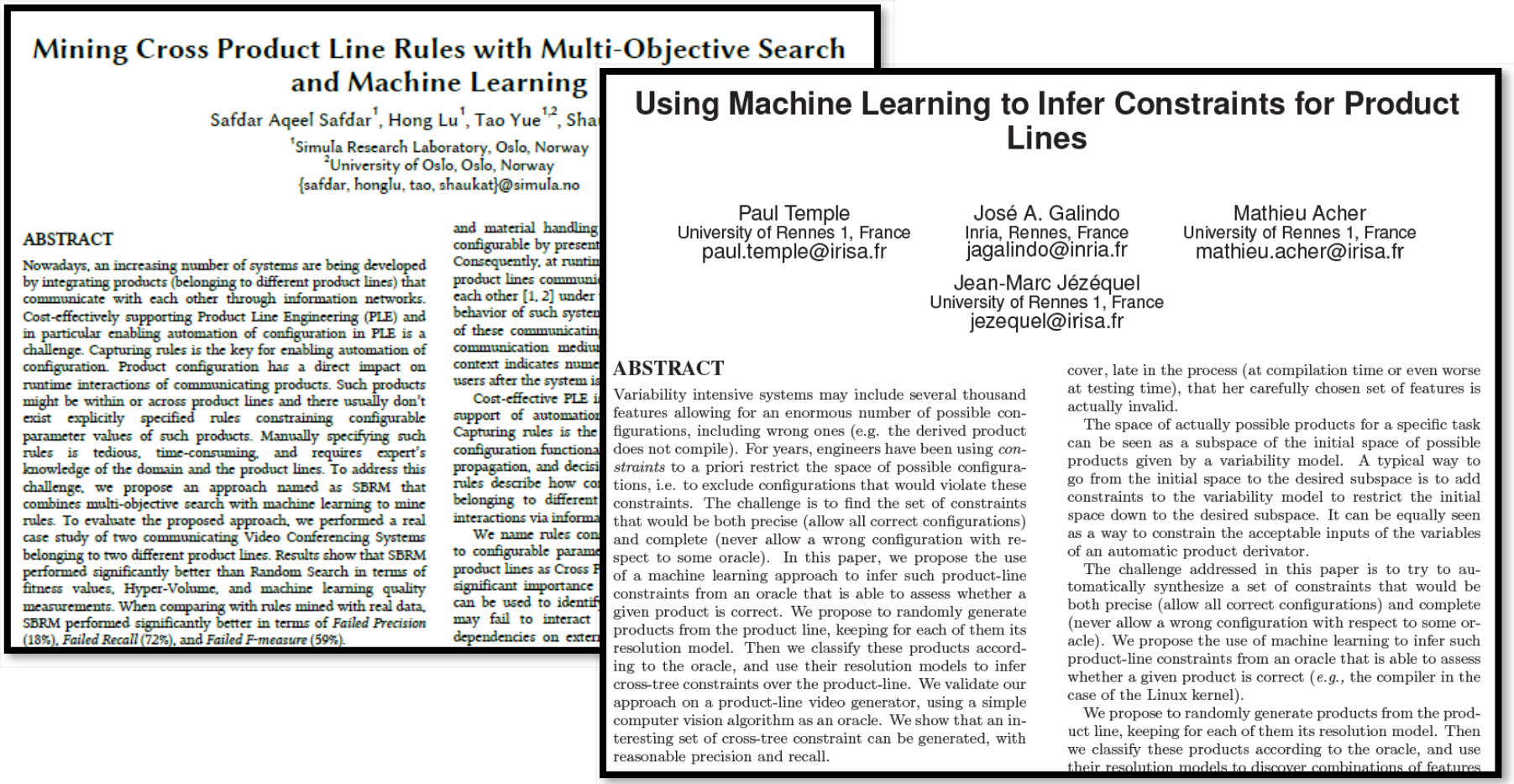
ABSTRACT

Software Product Lines (SPLs) are highly configurable systems. This raises the challenge to find optimal performing configurations for an anticipated workload. As SPL configuration spaces are huge, it is infeasible to benchmark all configurations to find an optimal one. Prior work focused on building performance models to predict and optimize SPL configurations. Instead, we randomly sample and recursively search a configuration space directly to find near-optimal configurations without constructing a prediction model. Our algorithms are simpler and have higher accuracy and efficiency.

samples can be taken for performance models to have acceptable accuracy?

This paper focuses on a fundamental problem in SPLs to find acceptable configurations whose performance is near-optimal. We do ***not*** create a performance prediction model, which then requires an optimizer (*e.g.* using a genetic algorithm [27]) to find good configurations. Instead, we use BDDs to count the number of valid configurations in a configuration space, thereby enabling true random sampling of the space. Doing so allows us to prove theoretically tight bounds on sampling results. Further, we identify features that are statistically certain to improve or degrade program performance

Thema 2 – Using ML to Infer Constraints



Mining Cross Product Line Rules with Multi-Objective Search and Machine Learning

Safdar Aqeel Safdar[1], Hong Lu[1], Tao Yue[1,2], Sha
[1]Simula Research Laboratory, Oslo, Norway
[2]University of Oslo, Oslo, Norway
{safdar, honglu, tao, shaukat}@simula.no

ABSTRACT

Nowadays, an increasing number of systems are being developed by integrating products (belonging to different product lines) that communicate with each other through information networks. Cost-effectively supporting Product Line Engineering (PLE) and in particular enabling automation of configuration in PLE is a challenge. Capturing rules is the key for enabling automation of configuration. Product configuration has a direct impact on runtime interactions of communicating products. Such products might be within or across product lines and there usually don't exist explicitly specified rules constraining configurable parameter values of such products. Manually specifying such rules is tedious, time-consuming, and requires expert's knowledge of the domain and the product lines. To address this challenge, we propose an approach named as SBRM that combines multi-objective search with machine learning to mine rules. To evaluate the proposed approach, we performed a real case study of two communicating Video Conferencing Systems belonging to two different product lines. Results show that SBRM performed significantly better than Random Search in terms of fitness values, Hyper-Volume, and machine learning quality measurements. When comparing with rules mined with real data, SBRM performed significantly better in terms of *Failed Precision* (18%), *Failed Recall* (72%), and *Failed F-measure* (59%).

and material handling
configurable by present
Consequently, at runtim
product lines communi
each other [1, 2] under
behavior of such syste
of these communicatin
communication mediu
context indicates nume
users after the system is

Cost-effective PLE i
support of automatio
Capturing rules is the
configuration functiona
propagation, and decisi
rules describe how co
belonging to different
interactions via informa

We name rules con
to configurable parame
product lines as Cross P
significant importance
can be used to identif
may fail to interact
dependencies on exter

Using Machine Learning to Infer Constraints for Product Lines

Paul Temple
University of Rennes 1, France
paul.temple@irisa.fr

José A. Galindo
Inria, Rennes, France
jagalindo@inria.fr

Mathieu Acher
University of Rennes 1, France
mathieu.acher@irisa.fr

Jean-Marc Jézéquel
University of Rennes 1, France
jezequel@irisa.fr

ABSTRACT

Variability intensive systems may include several thousand features allowing for an enormous number of possible configurations, including wrong ones (e.g. the derived product does not compile). For years, engineers have been using *constraints* to a priori restrict the space of possible configurations, i.e. to exclude configurations that would violate these constraints. The challenge is to find the set of constraints that would be both precise (allow all correct configurations) and complete (never allow a wrong configuration with respect to some oracle). In this paper, we propose the use of a machine learning approach to infer such product-line constraints from an oracle that is able to assess whether a given product is correct. We propose to randomly generate products from the product line, keeping for each of them its resolution model. Then we classify these products according to the oracle, and use their resolution models to infer cross-tree constraints over the product-line. We validate our approach on a product-line video generator, using a simple computer vision algorithm as an oracle. We show that an interesting set of cross-tree constraint can be generated, with reasonable precision and recall.

cover, late in the process (at compilation time or even worse at testing time), that her carefully chosen set of features is actually invalid.

The space of actually possible products for a specific task can be seen as a subspace of the initial space of possible products given by a variability model. A typical way to go from the initial space to the desired subspace is to add constraints to the variability model to restrict the initial space down to the desired subspace. It can be equally seen as a way to constrain the acceptable inputs of the variables of an automatic product derivator.

The challenge addressed in this paper is to try to automatically synthesize a set of constraints that would be both precise (allow all correct configurations) and complete (never allow a wrong configuration with respect to some oracle). We propose the use of machine learning to infer such product-line constraints from an oracle that is able to assess whether a given product is correct (*e.g.*, the compiler in the case of the Linux kernel).

We propose to randomly generate products from the product line, keeping for each of them its resolution model. Then we classify these products according to the oracle, and use their resolution models to discover combinations of features

Thema 3 – Performance Influence Models

Tradeoffs in modeling performance of highly configurable software systems

Sergiy Kolesnikov[1] · Norbert Siegmund[2] · Christian Kästner[3] · Alexande

Received: 25 July 2016 / Revised: 19 October 2017 / Accepted: 12 January 2018 / Published onlin

Abstract
Modeling the performance of a highly configurable software system require
options and their interactions on the system's performance. *Performance-influen*
this way the performance behavior of a configurable system as a whole. To
model should have a low prediction error, small model size, and reasonable com
among these properties, optimizing for one property may negatively influence
these tradeoffs manifest themselves in practice, that is, whether a large config
with large models and significant resource investment. By means of 10 real-wo
domains, we have systematically studied the tradeoffs between the three prope
between prediction error and model size and between prediction error and cor
can learn accurate and small models in reasonable time, so that one performa
such as program comprehension and performance prediction. We further inve
marginal. We found that interactions among four or more configuration option
error and that ignoring them when learning a performance-influence model
time, while keeping the model small without considerably increasing the pre
new sampling and learning techniques as they can focus on specific regions o
between accuracy and effort. We further analyzed the causes for the configur
observed influences on the systems' performance. We were able to identify s
dominant configuration options and data pipelines, that explain the influences
interactions, and give further insights into the domain of highly configurable s

Performance-Influence Models for Highly Configurable Systems

Norbert Siegmund†, Alexander Grebhahn†, Sven Apel†, Christian Kästner‡
†University of Passau, Germany ‡Carnegie Mellon University, USA

ABSTRACT

Almost every complex software system today is configurable. While configurability has many benefits, it challenges performance prediction, optimization, and debugging. Often, the influences of individual configuration options on performance are unknown. Worse, configuration options may interact, giving rise to a configuration space of possibly exponential size. Addressing this challenge, we propose an approach that derives a *performance-influence model* for a given configurable system, describing all *relevant* influences of configuration options and their interactions. Our approach combines machine-learning and sampling heuristics in a novel way. It improves over standard techniques in that it (1) represents influences of options and their interactions explicitly (which eases debugging), (2) smoothly integrates binary and numeric configuration options for the first time, (3) incorporates domain knowledge, if available (which eases learning and increases accuracy), (4) considers complex constraints among options, and (5) systematically reduces the solution space to a tractable size. A series of experiments demonstrates the feasibility of our approach in terms of the accuracy of the models learned as well as the accuracy of the performance predictions one can make with them.

among configuration options make it difficult to find a configuration that performs as desired, with the consequence that many users stick to default configurations or only try changing an option here or there. This way, the significant optimization potential already built in many of our modern software systems remains untapped. Even domain experts and the developers themselves often do not (fully) understand the performance influences of all configuration options and their combined influence when they interact.

Our goal is to build performance-influence models (and models of other measurable quality attributes, such as energy consumption) that describe how configuration options and their interactions influence the performance of a system (e.g., throughput or execution time of a benchmark). Performance-influence models are meant to ease *understanding, debugging, and optimization* of highly configurable software systems. For example, an end user may use an optimizer to identify the best performing configuration under certain constraints (e.g., encryption needs to be enabled) from the model; a database administrator may use it to determine the influence of certain configuration options and how they interact; and a developer may compare an inferred performance-influence model with her own mental model to check whether the system behaves as expected.

Thema 4 – SATzilla

SATzilla: Portfolio-based Algorithm Selection for SAT

Lin Xu XULIN730@CS.UBC.CA
Frank Hutter HUTTER@CS.UBC.CA
Holger H. Hoos HOOS@CS.UBC.CA
Kevin Leyton-Brown KEVINLB@CS.UBC.CA
Department of Computer Science
University of British Columbia
201-2366 Main Mall, BC V6T 1Z4, CANADA

Abstract

It has been widely observed that there is no single "dominant" SAT solver; instead, different solvers perform best on different instances. Rather than following the traditional approach of choosing the best solver for a given class of instances, we advocate making this decision online on a per-instance basis. Building on previous work, we describe `SATzilla`, an automated approach for constructing per-instance algorithm portfolios for SAT that use so-called empirical hardness models to choose among their constituent solvers. This approach takes as input a distribution of problem instances and a set of component solvers, and constructs a portfolio optimizing a given objective function (such as mean runtime, percent of instances solved, or score in a competition). The excellent performance of `SATzilla` was independently verified in the 2007 SAT Competition, where our `SATzilla07` solvers won three gold, one silver and one bronze medal. In this article, we go well beyond `SATzilla07` by making the portfolio construction scalable and completely automated, and improving it by integrating local search solvers as candidate solvers, by predicting performance score instead of runtime, and by using hierarchical hardness models that take into account different types of SAT instances. We demonstrate the effectiveness of these new techniques in extensive experimental results on data sets including instances from the most recent SAT competition.

Thema 5 – Transferring Performance Models

Transfer Learning for Improving Model Predictions in Highly Configurable Software

Pooyan Jamshidi, Miguel Velez, Christian Kästner
Carnegie Mellon University, USA
{pjamshid,mvelezce,kaestner}@cs.cmu.edu

Norbert Siegmund
Bauhaus-University Weimar, Germany
norbert.siegmund@uni-weimar.de

***Abstract*—Modern software systems are built to be used in dynamic environments using configuration capabilities to adapt to changes and external uncertainties. In a self-adaptation context, we are often interested in reasoning about the performance of the systems under different configurations. Usually, we learn a black-box model based on real measurements to predict the performance of the system given a specific configuration. However, as modern systems become more complex, there are many configuration parameters that may interact and we end up learning an exponentially large configuration space. Naturally, this does not scale when relying on real measurements in the actual changing environment. We propose a different solution: Instead of taking the measurements from the real system, we learn the model using samples from other sources, such as simulators that approximate performance of the real system at low cost. We define a cost model that transform the traditional view of model learning into a multi-objective problem that not only takes into account model accuracy but also measurements effort as well. We evaluate our cost-aware transfer learning solution using real-world configurable software including (i) a robotic system, (ii) 3 different stream processing applications, and (iii) a NoSQL database system. The experimental results demonstrate that our approach can achieve (a) a high prediction accuracy, as well as (b) a high model reliability.**

***Index Terms*—highly configurable software, machine learning, model learning, model prediction, transfer learning**

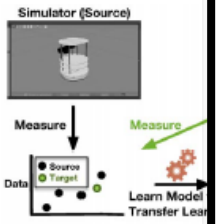


Fig. 1: Transfer learning for p

order to identify the best perfor
when it goes from indoor to an

Typically, we learn a perfor
figurable system by measuring
selected by some sampling stra
performance of a given syste
configurations and learn how
their interactions affect perfor
of learning from real systems
software application, is a diffi
(i) environmental changes (e

Transferring Performance Prediction Models Across Different Hardware Platforms

Pavel Valov
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
pvalov@uwaterloo.ca

Jean-Christophe Petkovich
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
j2petkovich@uwaterloo.ca

Jianmei Guo*
East China University of Science and Technology
130 Meilong Road
Shanghai, China
gjm@ecust.edu.cn

Sebastian Fischmeister
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
sfischme@uwaterloo.ca

Krzysztof Czarnecki*
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
kczarnec@gsd.uwaterloo.ca

ABSTRACT

Many software systems provide configuration options relevant to users, which are often called features. Features influence functional properties of software systems as well as non-functional ones, such as performance and memory consumption. Researchers have successfully demonstrated the correlation between feature selection and performance. However, the generality of these performance models across different hardware platforms has not yet been evaluated.

We propose a technique for enhancing generality of performance models across different hardware environments using linear transformation. Empirical studies on three real-world software systems show that our approach is computationally efficient and can achieve high accuracy (less than 10% mean relative error) when predicting system performance across 23 different hardware platforms. Moreover, we investigate why the approach works by comparing performance distributions of systems and structure of performance models across different platforms.

1. INTRODUCTION

Many software systems provide *configuration options*. These configuration options usually have a direct influence on the functional behaviour of target software systems. Some configuration options may impact systems' non-functional properties, such as response time, memory consumption and throughput. Configuration options that are relevant to users are usually called *features* [5], and a particular selection of features defines a system *configuration*.

Performance prediction of configurable software systems is a highly-researched topic [5, 6, 7, 8, 17, 18, 20, 21]. For example, Guo et al. [5] predicted a system configuration's performance by using regression trees based on small random samples of measured configurations. However, none of the previous work studied whether or not it is possible to transfer performance prediction models for configurable software systems across different hardware platforms.

The need for transferring performance prediction models occurs in many application scenarios. For example, a user of

Thema 6 – Combinatorial BLAS

The Combinatorial BLAS: Design, Implementation, and Applications *†

Aydın Buluç ‡§
High Performance Computing Research
Lawrence Berkeley National Laboratory
1 Cyclotron Road, Berkeley, CA 94720
abuluc@lbl.gov

John R. Gilbert
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106-5110
gilbert@cs.ucsb.edu

Abstract

This paper presents a scalable high-performance software library to be used for graph analysis and data mining. Large combinatorial graphs appear in many applications of high-performance computing, including computational biology, informatics, analytics, web search, dynamical systems, and sparse matrix methods. Graph computations are difficult to parallelize using traditional approaches due to their irregular nature and low operational intensity. Many graph computations, however, contain sufficient coarse grained parallelism for thousands of processors, which can be uncovered by using the right primitives.

We describe the parallel Combinatorial BLAS, which consists of a small but powerful set of linear algebra primitives specifically targeting graph and data mining applications. We provide an extendible library interface and some guiding principles for future development. The library is evaluated using two important graph algorithms, in terms of both performance and ease-of-use. The scalability and raw performance of the example applications, using the combinatorial BLAS, are unprecedented on distributed memory clusters.

Thema 7 – Memory Error Detection 1

AddressSanitizer: A Fast Address Sanity Checker

Konstantin Serebryany, Derek Bruening, Alexander Potapenko, Dmitry Vyukov
Google
{*kcc,bruening,glider,dvyukov*}*@google.com*

Abstract

Memory access bugs, including buffer overflows and uses of freed heap memory, remain a serious problem for programming languages like C and C++. Many memory error detectors exist, but most of them are either slow or detect a limited set of bugs, or both.

This paper presents AddressSanitizer, a new memory error detector. Our tool finds out-of-bounds accesses to heap, stack, and global objects, as well as use-after-free bugs. It employs a specialized memory allocator and code instrumentation that is simple enough to be implemented in any compiler, binary translation system, or even in hardware.

AddressSanitizer achieves efficiency without sacrificing comprehensiveness. Its average slowdown is just

zones around stack and global
and underflows. The current i
the LLVM [4] compiler infras
brary replaces `malloc`, `free`
ates poisoned redzones arou
delays the reuse of freed hea
reporting.

1.1 Contributions

In this paper we:

- show that a memory erro
comprehensiveness of sh
lower overhead than the

Enhanced Memory Corruption Detection in C/C++ Programs

Ching-Yi Lin
jennifer710846@gmail.com
Master Degree of Computer Security
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

Wuu Yang
wuuyang@cs.nycu.edu.tw
Department of Computer Science
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

ABSTRACT

Out-of-bound memory accesses, which often occur in programs written in unsafe languages such as C or C++, cause severe troubles. Though there are many useful tools aiming at this problem, we report a new tool, called *mcds*, for detecting spatial and temporal memory corruptions in x86-64 ELF binary. Mcds allocates each memory object to a separate virtual page. The rest is left blank. Due to a facility in the memory management library, we can set up memory protection so that accessing the "blank" part of a virtual page causes a hardware trap. Because it is a hardware trap, there is little run-time overhead. In order to save memory space, we may squeeze several virtual pages into a single physical page. Our first experimental result is that mcds can find all the bugs in the Firefox 78 package, the Chrome package and the PHP7.0 package that are recorded on the CVE Details website. Furthermore, mcds can detect three classes of memory corruptions that are beyond the capability of the current AddressSanitizer (Asan). Then we compare the time for compilation and fuzzing tests. The fuzzing test is done with AFL++ fuzzer on Ubuntu 22.04 LTS with Intel i5-9600K chip. According to our experimental results, mcds shows approximately 6x speedup in fuzzing tests against AddressSanitizer. There is not significant difference between compiling the source with AddressSanitizer or with mcds, though both of them result in 2x slowdown compared with compilation without a sanitizer.

1 INTRODUCTION

Memory corruption is one of the most common vulnerabilities for programs that are written in memory-unsafe languages. Especially in the large-scale projects such as OS kernels, libraries or browser packages, memory corruptions may cause catastrophic impacts.

There are several tools for detecting memory corruptions. LowFat Pointer [11], AdressSanitizer (i.e., Asan) [17], and MEDS [9] are static detection tools, all of which are based on the LLVM toolchain. The source code is needed for debugging purposes in these static tools. The dynamic detection tools such as Valgrind [16] and Intel PIN [15] perform detection directly on binaries while execution. However, the above tools still meet some problems. For example, LowFat Pointer only detects spatial memory errors, but it causes many kinds of caveats [7]. LowFat optimizes the out-of-bound stack access detection with global sharable memory mapping, which sometimes crashes the program when `fork`, `clone` or some `clone`-based function interceptors in Lowfat are invoked. Therefore, Lowfat is not recommended for debugging multiprocess programs. [6] Asan and Valgrind, which are the most well-known tools currently, also meet their difficulties in several scenarios of memory corruptions. Valgrind performs fine-grained examination with dynamic instrumentation for detecting temporal memory corruptions. Valgrind re-executes the process shadowed with UCode to clarify whether there is any memory corruption. Valgrind results in huge time and space overhead. Furthermore, detecting memory errors

Thema 8 – Memory Error Detection 2

Finding Unstable Code via Compiler-Driven Differential Testing

Shaohua Li
shaohua.li@inf.ethz.ch
ETH Zurich
Switzerland

Zhendong Su
zhendong.su@inf.ethz.ch
ETH Zurich
Switzerland

ABSTRACT

Unstable code refers to code that has inconsistent or unstable runtime semantics due to undefined behavior (UB) in the program. Compilers exploit UB by assuming that UB never occurs, which allows them to generate efficient but potentially semantically inconsistent binaries. Practitioners have put great research and engineering effort into designing dynamic tools such as sanitizers for frequently occurring UBs. However, it remains a big challenge how to detect UBs that are beyond the reach of current techniques.

In this paper, we introduce compiler-driven differential testing (CompDiff), a simple yet effective approach for finding unstable code in C/C++ programs. CompDiff relies on the fact that when compiling unstable code, different compiler implementations may produce semantically inconsistent binaries. Our main approach is to examine the outputs of different binaries on the same input. Discrepancies in outputs may signify the existence of unstable code. To detect unstable code in real-world programs, we also integrate CompDiff into AFL++, the most widely-used and actively-maintained general-purpose fuzzer.

Despite its simplicity, CompDiff is effective in practice: on the Juliet benchmark programs, CompDiff uniquely detected 1,409 bugs compared to sanitizers; on 23 popular open-source C/C++ projects, CompDiff-AFL++ uncovered 78 new bugs, 52 of which have been fixed by developers and 36 cannot be detected by sanitizers. Our evaluation also reveals the fact that CompDiff is not designed to replace current UB detectors but to complement them.

```
1  /* dump a chunk of buffer*/
2  int dump_data (int offset, int len) {
3      char *data = /* buffer head */;
4      int size = /* size of buffer*/;
5      if (offset + len > size ||
6          offset < 0 || len < 0) {
7          return -1;
8      }
9      if (offset + len < offset) {
10         return -1;
11     }
12     /* dump from data+offset
13        to data+offset+len */
14     dump(data+offset, len);
15     return 0;
16 }
```

Listing 1: The second if guard in line 9 got optimized away by clang because it would only be evaluated to true when a signed integer overflow happened.

1 INTRODUCTION

Some programming languages such as C/C++ designate a set of

Thema 9 – Fuzzing 1

Magma: A Ground-Truth Fuzzing Benchmark

AHMAD HAZIMEH, EPFL, Switzerland
ADRIAN HERRERA, ANU & DST, Australia
MATHIAS PAYER, EPFL, Switzerland

High scalability and low running costs have made fuzz testing the de facto standard for discovering software bugs. Fuzzing techniques are constantly being improved in a race to build the ultimate bug-finding tool. However, while fuzzing excels at finding bugs in the wild, evaluating and comparing fuzzer performance is challenging due to the lack of metrics and benchmarks. For example, crash count—perhaps the most commonly-used performance metric—is inaccurate due to imperfections in deduplication techniques. Additionally, the lack of a unified set of targets results in ad hoc evaluations that hinder fair comparison.

We tackle these problems by developing *Magma*, a ground-truth fuzzing benchmark that enables uniform fuzzer evaluation and comparison. By introducing *real* bugs into *real* software, Magma allows for the realistic evaluation of fuzzers against a broad set of targets. By instrumenting these bugs, Magma also enables the collection of bug-centric performance metrics independent of the fuzzer. Magma is an open benchmark consisting of seven targets that perform a variety of input manipulations and complex computations, presenting a challenge to state-of-the-art fuzzers.

We evaluate seven widely-used mutation-based fuzzers (AFL, AFLFast, AFL++, FairFuzz, MOpt-AFL, honggfuzz, and SymCC-AFL) against Magma over 200,000 CPU-hours. Based on the number of bugs reached, triggered, and detected, we draw conclusions about the fuzzers' exploration and detection capabilities. This provides insight into fuzzer performance evaluation, highlighting the importance of ground truth in performing more accurate and meaningful evaluations.

Thema 10 – Fuzzing 2

FormatFuzzer: Effective Fuzzing of Binary File Formats

RAFAEL DUTRA, CISPA Helmholtz Center for Information Security, Germany
RAHUL GOPINATH, University of Sydney, Australia
ANDREAS ZELLER, CISPA Helmholtz Center for Information Security, Germany

Effective fuzzing of programs that process structured binary inputs, such as multimedia files, is a challenging task, since those programs expect a very specific input format. Existing fuzzers, however, are mostly *format-agnostic,* which makes them versatile, but also ineffective when a specific format is required.

We present FormatFuzzer, a generator for *format-specific fuzzers.* FormatFuzzer takes as input a *binary template* (a format specification used by the 010 Editor) and compiles it into C++ code that acts as *parser, mutator,* and highly efficient *generator* of inputs conforming to the rules of the language.

The resulting format-specific fuzzer can be used as a standalone producer or mutator in black-box settings, where no guidance from the program is available. In addition, by providing mutable *decision seeds,* it can be easily *integrated* with arbitrary *format-agnostic fuzzers* such as AFL to make them format-aware. In our evaluation on complex formats such as MP4 or ZIP, FormatFuzzer showed to be a highly effective producer of valid inputs that also detected previously unknown memory errors in `ffmpeg` and `timidity`.

Thema 11 – Energy Games

Energy Games in Multiweighted Automata*

Uli Fahrenberg[1], Line Juhl[2], Kim G. Larsen[2], and Jiří Srba[2**]

[1] INRIA/IRISA, Rennes Cedex, France
ulrich.fahrenberg@irisa.fr
[2] Aalborg University, Department of Computer Science, Denmark
{linej,kgl,srba}@cs.aau.dk

Abstract. Energy games have recently attracted a lot of attention. These are games played on finite weighted automata and concern the existence of infinite runs subject to boundary constraints on the accumulated weight, allowing e.g. only for behaviours where a resource is always available (nonnegative accumulated weight), yet does not exceed a given maximum capacity. We extend energy games to a multiweighted and parameterized setting, allowing us to model systems with multiple quantitative aspects. We present reductions between Petri nets and multiweighted automata and among different types of multiweighted automata and identify new complexity and (un)decidability results for both one- and two-player games. We also investigate the tractability of an extension of multiweighted energy games in the setting of timed automata.

Thema 12 – Database Systems and Business Processes (1)

A Decade of Business Process Management Conferences: Personal Reflections on a Developing Discipline

Wil M.P. van der Aalst

Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, The Netherlands.
www.vdaalst.com

Abstract. The Business Process Management (BPM) conference series celebrates its tenth anniversary. This is a nice opportunity to reflect on a decade of BPM research. This paper describes the history of the conference series, enumerates twenty typical BPM use cases, and identifies six key BPM concerns: process modeling languages, process enactment infrastructures, process model analysis, process mining, process flexibility, and process reuse. Although BPM matured as a research discipline, there are still various important open problems. Moreover, despite the broad interest in BPM, the adoption of state-of-the-art results by software vendors, consultants, and end-users leaves much to be desired. Hence, the BPM discipline should not shy away from the key challenges and set clear targets for the next decade.

Thema 12 – Database Systems and Business Processes (2)

A Quest for Beauty and Wealth (or, Business Processes for Database Researchers)

Daniel Deutch
Ben Gurion University
deutchd@cs.bgu.ac.il

Tova Milo
Tel Aviv University
milo@cs.tau.ac.il

ABSTRACT

While classic data management focuses on the data itself, research on Business Processes considers also the *context* in which this data is generated and manipulated, namely the processes, the users, and the goals that this data serves. This allows the analysts a better perspective of the organizational needs centered around the data. As such, this research is of fundamental importance.

Much of the success of database systems in the last decade is due to the beauty and elegance of the relational model and its declarative query languages, combined with a rich spectrum of underlying evaluation and optimization techniques, and efficient implementations. This, in turn, has lead to an economic wealth for both the users and vendors of database systems. Similar beauty and wealth are sought for in the context of Business Processes. Much like the case for traditional database research, elegant modeling and rich underlying technology are likely to bring economic wealth for the Business Process owners and their users; both can benefit from easy formulation and analysis of the processes. While there have been many important advances in this research in recent years, there is still much to be desired: specifically, there have been many works that focus on the processes behavior (flow), and many that focus on its data , but only very few works have dealt with both. We will discuss here the important advantages of a holistic flow-and-data framework for Business Processes, the progress towards such a framework, and highlight the current gaps and research directions.

Thema 12 – Database Systems and Business Processes (3)

Automatic Verification of Database-Driven Systems: A New Frontier

Victor Vianu*
U.C. San Diego
La Jolla, CA 92093
USA

ABSTRACT

We describe a novel approach to verification of software systems centered around an underlying database. Instead of applying general-purpose techniques with only partial guarantees of success, it identifies restricted but reasonably expressive classes of applications and properties for which sound and complete verification can be performed in a fully automatic way. This leverages the emergence of high-level specification tools for database-centered applications that not only allow fast prototyping and improved programmer productivity but, as a side effect, provide convenient targets for automatic verification. We present theoretical and practical results on verification of database-driven systems. The results are quite encouraging and suggest that, unlike arbitrary software systems, significant classes of database-driven systems may be amenable to automatic verification. This relies on a novel marriage of database and model checking techniques, of relevance to both the database and the computer aided verification communities.

Thema 12 – Database Systems and Business Processes (4)

E-Services: A Look Behind the Curtain

Richard Hull Michael Benedikt
Bell Labs Research
Lucent Technologies
Murray Hill Lisle
New Jersey, USA Illinois, USA
{hull,benedikt}@lucent.com

Vassilis Christophides
Institute for of Computer Science
Foundation for Research and
Technology-Hellas (FORTH)
Vassilika Vouton, Crete
christop@ics.forth.gr

Jianwen Su
Computer Science Department
University of California
Santa Barbara
California, USA
su@cs.ucsb.edu

ABSTRACT

The emerging paradigm of electronic services promises to bring to distributed computation and services the flexibility that the web has brought to the sharing of documents. An understanding of fundamental properties of e-service composition is required in order to take full advantage of the paradigm. This paper examines proposals and standards for e-services from the perspectives of XML, data management, workflow, and process models. Key areas for study are identified, including behavioral service signatures, verification and synthesis techniques for composite services, analysis of service data manipulation commands, and XML analysis applied to service specifications. We give a sample of the relevant results and techniques in each of these areas.

Thema 12 – Database Systems and Business Processes (5)

Model Checking for Database Theoreticians

Moshe Y. Vardi[1⋆]

Rice University
Houston, TX, USA
vardi@cs.rice.edu
http://www.cs.rice/edu/~vardi

Abstract. Algorithmic verification is one of the most successful applications of automated reasoning in computer science. In algorithmic verification one uses algorithmic techniques to establish the correctness of the system under verification with respect to a given property. Model checking is an algorithmic-verification technique that is based on a small number of key ideas, tying together graph theory, automata theory, and logic. In this self-contained talk I will describe how this "holy trinity" gave rise to algorithmic-verification tools, and discuss its applicability to database verification.

Themenverteilung

- Bachelor: ein Thema. Master: zwei Themen.
- Bis **Freitag, den 12.04. 12 Uhr**:
  - Melden Sie sich in Unisono zur Studienleistung an.
  - Wählen Sie drei bevorzugte Themen aus.
  - Senden Sie die Liste der drei Themen und ob Sie Proseminar oder Seminar studieren an Robert.Mueller@uni-siegen.de.
  - Wir bemühen uns, alle Themenwünsche zu berücksichtigen.
- Informationen zum ersten Meilenstein erhalten Sie danach per E-Mail von Ihrem Themenbetreuer.

Fragen