Introduction to Artifact Evaluation

Neea Rusch
Augusta University

3 April 2024

What is an Artifact?

An artifact is a supplement that extends beyond a scientific paper and supports the claims or results of that paper.

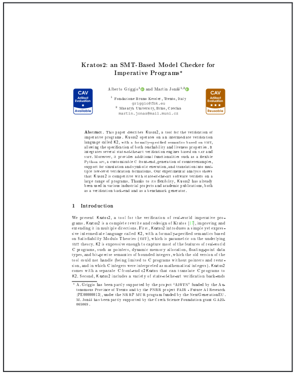
Artifact may contain: software, mechanized proofs, test suites, data sets, benchmarks, video of a difficult/impossible-to-share system in use, hardware, or any other artifact described in a paper.

An artifact captures a point-in-time matching the paper. It should be packaged for long-term preservation to facilitate future research.

Artifact Evaluation Motivations

- Encourage and support authors to provide supplements to papers
- Help future researchers to more effectively build on and compare with previous work
- Validate claims and results presented in a paper
- Reward authors who put in effort to create useful artifacts
- Recognize the effort to release usable software systems

Papers with artifacts are recognized with badges.



CAV Artifact Evaluation Available

CAV Artifact Evaluation Reusable

Kratos2: an SMT-Based Model Checker for Imperative Programs*

Alberto Griggio[1] and Martin Jonáš[1,2]

[1] Fondazione Bruno Kessler, Trento, Italy
griggio@fbk.eu
[2] Masaryk University, Brno, Czechia
martin.jonas@mail.muni.cz

Abstract. This paper describes Kratos2, a tool for the verification of imperative programs. Kratos2 operates on an intermediate verification language called K2, with a formally-specified semantics based on SMT, allowing the specification of both reachability and liveness properties. It integrates several state-of-the-art verification engines based on SAT and SMT. Moreover, it provides additional functionalities such as a flexible Python API, a customizable C front-end, generation of counterexamples, support for simulation and symbolic execution, and translation into multiple low-level verification formalisms. Our experimental analysis shows that Kratos2 is competitive with state-of-the-art software verifiers on a large range of programs. Thanks to its flexibility, Kratos2 has already been used in various industrial projects and academic publications, both as a verification back-end and as a benchmark generator.

1 Introduction

We present Kratos2, a tool for the verification of real-world imperative programs. Kratos2 is a complete rewrite and redesign of Kratos [17], improving and extending it in multiple directions. First, Kratos2 introduces a simple yet expressive intermediate language called K2, with a formally-specified semantics based on Satisfiability Modulo Theories (SMT), which is parametric on the underlying SMT theory. K2 is expressive enough to capture most of the features of real-world C programs, such as pointers, dynamic memory allocation, floating-point data types, and bit-precise semantics of bounded integers, which the old version of the tool could not handle (being limited to C programs without pointers and recursion, and in which C integers were interpreted as mathematical integers). Kratos2 comes with a separate C front-end c2Kratos that can translate C programs to K2. Second, Kratos2 includes a variety of state-of-the-art verification back-ends

* A. Griggio has been partly supported by the project "AI@TN" funded by the Autonomous Province of Trento and by the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU. M. Jonáš has been partly supported by the Czech Science Foundation grant GA23-06506S.

Merchandiser: Data Placement on Heterogeneous Memory for Task-Parallel HPC Applications with Load-Balance Awareness

Zhen Xie
zhen.xie@anl.gov
University of California, Merced
Argonne National Laboratory

Jie Liu
jliu279@ucmerced.edu
University of California, Merced

Jiajia Li
jiajia.li@ncsu.edu
North Carolina State University

Dong Li
dli35@ucmerced.edu
University of California, Merced

Abstract

The emergence of heterogeneous memory (HM) provides a cost-effective and high-performance solution to memory-consuming HPC applications. Deciding the placement of data objects on HM is critical for high performance. We reveal a performance problem related to data placement on HM. The problem is manifested as load imbalance among tasks in task-parallel HPC applications. The root of the problem comes from being unaware of parallel-task semantics and an incorrect assumption that bringing frequently accessed pages to fast memory always leads to better performance. To address this problem, we introduce a load balance-aware page management system, named *Merchandiser*. Merchandiser introduces task semantics during memory profiling, rather than being application-agnostic. Using the limited task semantics, Merchandiser effectively sets up coordination among tasks on the usage of HM to finish *all* tasks fast instead of only considering any individual task. **Merchandiser is highly automated to enable high usability. Evaluating with memory-consuming HPC applications, we show that Merchandiser reduces load imbalance and leads to an average of 17.1% and 15.4% (up to 26.0% and 23.2%) performance improvement, compared with a hardware-based solution and an industry-quality software-based solution.**

CCS Concepts: • **Computer systems organization → Heterogeneous (hybrid) systems**; • **Theory of computation → Parallel computing models**; • **Hardware → Non-volatile memory**.

Keywords: Data Placement, Heterogeneous Memory, Parallel Computing, Load Balance

ACM Reference Format:
Zhen Xie, Jie Liu, Jiajia Li, and Dong Li. 2023. Merchandiser: Data Placement on Heterogeneous Memory for Task-Parallel HPC Applications with Load-Balance Awareness. In *The 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '23), February 25-March 1, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3572848.3577497

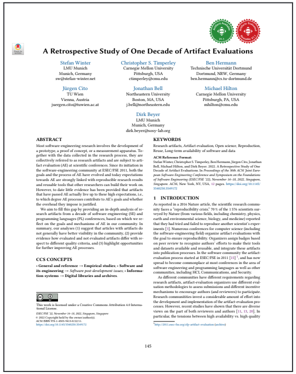
1 Introduction

Many high-performance computing (HPC) applications are becoming memory-consuming. For example, the density matrix renormalization group (DMRG) [6, 77], a numerical algorithm to obtain the low-energy physics of quantum many-body systems, can consume 1.271 TB memory in a single machine when solving the Hubbard 2D model at the scale of 320×320 [21, 46]. To meet memory requirements of those applications, the big memory system is emerging. An example of such a system is the Amazon EC2 High Memory Instance built upon eight NUMA nodes and providing up to 12 TB memory [31]. The big memory system is often heterogeneous, which means multiple memory components with different latency and bandwidth form the main memory.

HM raises a data placement problem. Because of small capacity of fast memory and relatively worse performance of slow memory, memory pages have to be allocated and migrated between fast and slow memories, such that most of memory accesses can happen in fast memory for high performance. It has been shown that some HPC applications can suffer from up to 5.7× performance loss (compared with using a fast memory-only solution) with suboptimal data placement on HM [61, 63, 67, 84].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PPoPP '23, February 25-March 1, 2023, Montreal, QC, Canada
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0015-6/23/02...$15.00
https://doi.org/10.1145/3572848.3577497

204



A Retrospective Study of One Decade of Artifact Evaluations

Stefan Winter, LMU Munich, Munich, Germany, sw@stefan-winter.net
Christopher S. Timperley, Carnegie Mellon University, Pittsburgh, USA, ctimperley@cmu.edu
Ben Hermann, Technische Universität Dortmund, Dortmund, NRW, Germany, ben.hermann@cs.tu-dortmund.de
Jürgen Cito, TU Wien, Vienna, Austria, juergen.cito@tuwien.ac.at
Jonathan Bell, Northeastern University, Boston, MA, USA, j.bell@northeastern.edu
Michael Hilton, Carnegie Mellon University, Pittsburgh, PA, USA, mhilton@cmu.edu
Dirk Beyer, LMU Munich, Munich, Germany, dirk.beyer@sosy-lab.org

ABSTRACT

Most software-engineering research involves the development of a prototype, a proof of concept, or a measurement apparatus. Together with the data collected in the research process, they are collectively referred to as research artifacts and are subject to artifact evaluation (AE) at scientific conferences. Since its initiation in the software-engineering community at ESEC/FSE 2011, both the goals and the process of AE have evolved and today expectations towards AE are strongly linked with reproducible research results and reusable tools that other researchers can build their work on. However, to date little evidence has been provided that artifacts that have passed AE actually live up to these high expectations, i.e., to which degree AE processes contribute to AE's goals and whether the overhead they impose is justified.

We aim to fill this gap by providing an in-depth analysis of research artifacts from a decade of software engineering (SE) and programming languages (PL) conferences, based on which we reflect on the goals and mechanisms of AE in our community. In summary, our analyses (1) suggest that articles with artifacts do not generally have better visibility in the community, (2) provide evidence how evaluated and not evaluated artifacts differ with respect to different quality criteria, and (3) highlight opportunities for further improving AE processes.

CCS CONCEPTS

• **General and reference → Empirical studies**; • **Software and its engineering →** *Software post-development issues*; • **Information systems → Digital libraries and archives**.

This work is licensed under a Creative Commons Attribution 4.0 International License.
ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9413-0/22/11.
https://doi.org/10.1145/3540250.3549172

KEYWORDS

Research artifacts, Artifact evaluation, Open science, Reproduction, Reuse, Long-term availability of software and data

ACM Reference Format:
Stefan Winter, Christopher S. Timperley, Ben Hermann, Jürgen Cito, Jonathan Bell, Michael Hilton, and Dirk Beyer. 2022. A Retrospective Study of One Decade of Artifact Evaluations. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3540250.3549172

1 INTRODUCTION

As reported in a 2016 Nature article, the scientific research community faces a "reproducibility crisis." 70 % of the 1 576 scientists surveyed by Nature (from various fields, including chemistry, physics, earth and environmental science, biology, and medicine) reported that they had tried and failed to reproduce another scientist's experiments [3]. Numerous conferences for computer science (including the software-engineering field) organize artifact evaluations with the goal to ensure reproducibility. Organizers assign badges based on peer review to recognize authors' efforts to make their tools and datasets available and reusable, and integrate these artifacts into publication processes. In the software community the artifact-evaluation process started at ESEC/FSE in 2011 [15] [1], and has now spread to become commonplace at most conferences in the area of software engineering and programming languages as well as other communities, including HCI, Communications, and Security.

As different communities have different requirements regarding research artifacts, artifact-evaluation organizers use different evaluation methodologies to assess submissions and different incentive mechanisms to encourage authors (and reviewers) to participate. Research communitities invest a considerable amount of effort into the development and implementation of the artifact-evaluation processes. However, recent studies have shown that there are diverse views on the part of both reviewers and authors [11, 13, 20]. In particular, the tensions between high availability vs. high quality

[1] http://2011.esec-fse.org/cfp-artifact-evaluation (archive)

145

Historical Background

Insufficient respect paid to the artifacts that back papers.

Areas so centered on software, models, and specifications should want to evaluate artifacts as part of the paper review process.

Not examining artifacts enables everything from mere sloppiness to, in extreme cases, dishonesty.

More subtly, it also imposes a subtle penalty on people who take the trouble to vigorously implement and test their ideas.

Source: artifact-eval.org/motivation

Historical Background

In 2011, Andreas Zeller, the program chair for ESEC/FSE, decided to institute a committee to address this problem. Andreas asked Carlo Ghezzi and Shriram Krishnamurthi to run this process.

Shriram had long wanted to create such a committee and call it the "Program Committee" (ha, ha). However, not only is that name taken, we also wanted to be open-minded to all sorts of artifacts that are not programs [...]. We therefore called this the ***Artifact Evaluation Committee (AEC)****.*

Source: artifact-eval.org/motivation

Artifact Evaluation Today

Software Engineering & Programming Languages conferences
ICSE, ESEC/FSE, ASE, ECOOP, ISSTA, OOPSLA, POPL, PLDI, ICFP, SAS, ESOP, TACAS, CAV…

Top SE/PL journals
TSE, TOSEM, EMSE, TOPLAS…

Systems research
SOSP, USENIX ATC, EuroSys, FAST, OSDI, SC…

High-Level Workflow

Common

$$\begin{tikzcd}
\text{paper submission} \arrow[d] \\
\text{paper (conditionally?) accepted} \arrow[d] \\
\text{artifact submission} \arrow[d] \\
\text{artifact decision} \arrow[d] \\
\text{camera-ready paper}
\end{tikzcd}$$

Alternative

$$\begin{tikzcd}
\text{paper submission} \arrow[d] & & \text{artifact submission} \arrow[d] \\
\text{paper decision} \arrow[dr] & & \text{artifact decision} \arrow[dl] \\
& \text{paper accepted?} \arrow[d] & \\
& \text{camera-ready paper} &
\end{tikzcd}$$

Artifact Evaluation Process

Time	Step	Responsible party
	Artifact submission	Authors
2–5 days	Bidding	AEC members
	Artifacts assigned (usually 2–3)	AEC chairs
1–2 weeks	**Phase 1:** Kick the tires	AEC members
1–2 weeks	Author responses, possible fixes	Authors
2–4 weeks	**Phase 2:** Full review	AEC members
3–7 days	Discussion and badging decisions	AEC members
	Decisions announced	AEC chairs

Expect an artifact to take on average 8h to review completely.

Badges

Artifacts are evaluated against badging criteria. The current commonly applied criteria is ACM Artifact Review and Badging policy v1.1.

https://www.acm.org/publications/policies/artifact-review-and-badging-current

Badges: Available

Author-created artifacts relevant to the paper have been placed on a publicly accessible *archival repository*.

A DOI or link to this repository along with a unique identifier for the object is provided.

Archival repositories: Zenodo, Figshare, Software Heritage, Dagstuhl Artifacts Series (DARTS),...

Badges: Artifacts Evaluated

Functional The artifacts associated with the research are found to be *documented*, *consistent*, *complete*, *exercisable*, and include appropriate evidence of verification and validation.

Reusable The artifacts have all the qualities of Functional level, but, in addition, they are very carefully documented and well-structured to the extent that reuse and repurposing is facilitated.

Badges: Results Validated

Reproduced The main results of the paper have been obtained in a subsequent study by a person or team other than the authors, *using, in part, artifacts provided by the author.*

Replicated The main results of the paper have been independently obtained in a subsequent study by a person or team other than the authors, *without the use of author-supplied artifacts.*

Artifact Evaluators

AEC members are usually senior graduate students, postdocs, or recent PhD graduates.

Among researchers, experienced graduate students are often in the best position to handle the diversity of systems expectations that the AEC will encounter.

Graduate students represent the future of the community, so involving them in the AEC process early will help push this process forward.

Source: https://pldi24.sigplan.org/track/pldi-2024-pldi-research-artifacts

Benefits of Participating in AECs

- Early experience in peer review process, learn how to write reviews
- Early access to cutting-edge works at top conferences
- Gain exposure to new research topics
- Develop intuition for what a top-conference publication requires
- Learn the artifact process and improve quality of own artifacts
- Start recognizing researchers, research trends, etc.
- Service experience for your CV

General Artifact Preparation Tips

- Make artifact claims explicit in the artifact readme
- Prepare a push-button/single command evaluation
- If paper has tables and figures of measurements, the artifact should support re-generating these
- Remember a license

General Artifact Preparation Tips

For long evaluations:

- Prepare a small evaluation ($\sim$10 min) + full evaluation
- Try include difficult cases in the small evaluation
- Provide time estimates of all long latency tasks

Artifact should be useful long-term ($\geq$ 10 years)

- Provide a container or VM that captures the expected environment
- Detail software dependencies, including versions
- Make artifact self-contained: avoid external references that may change or get deleted