Unicorn: Reasoning about Configurable System Performance through the Lens of Causality

Reproduktion und Evaluation

Alexander Vödisch
av21vupu@studserv.uni-leipzig.de

Abstract

In dieser Ausarbeitung werden die Ergebnisse von [ref] repliziert und evaluiert.

1 Einführung

Moderne Computersysteme sind oft aus einer Vielzahl von konfigurierbaren Komponenten aufgebaut, deren Zusammensetzung kritisch für die Systemperformance ist.[1] Da die optimale Konfiguration dieser Komponenten hardwareabhängig sein kann und insbesondere in Kombination zu Pipelines der Konfigurationsraum exponentiell wachsen kann, sind spezielle Algorithmen notwendig, um Nutzer solcher Systeme bei der Definition möglichst optimaler Konfigurationen zu unterstützen.

Das Ziel von UNICORN ist es, für konfigurierbare Systeme die optimalen Konfigurationen bezüglich eines vom Nutzer definierten Merkmals (z. B. Leistung, Energieverbrauch, Inferenzzeit) zu finden und im Zuge dessen einen kausalen Graphen zu generieren, der als Modell auch auf bisher unbekannte Harware übertragen werden kann.

Im Gegensatz zu Modellen, die beispielsweise auf Regression basieren und lediglich den Einfluss individueller Konfigurationsoptionen messen, bietet UNICORN den Vorteil, dass nicht die Korrelation von Konfigurationsoptionen zur Vorhersage von guten Konfigurationen, sondern die kausalen Zusammenhänge der einzelnen Optionen für die Entwicklung des Modells genutzt werden. Dadurch kann das Modell auch in unbekannten Environments genutzt werden, wo korrelationsbasierte Performance-Influence-Modelle oft nur unzuverlässige Vorhersagen treffen.

2 Funktionsweise

UNICORN ermöglicht sowohl das Debuggen als auch auch das Optimieren von Systemkonfigurationen. Hierzu definieren die Autoren ein kausales Modell das auf probabilistischen graphischen Modellen basiert. Diese kausalen Graphen bestehen aus

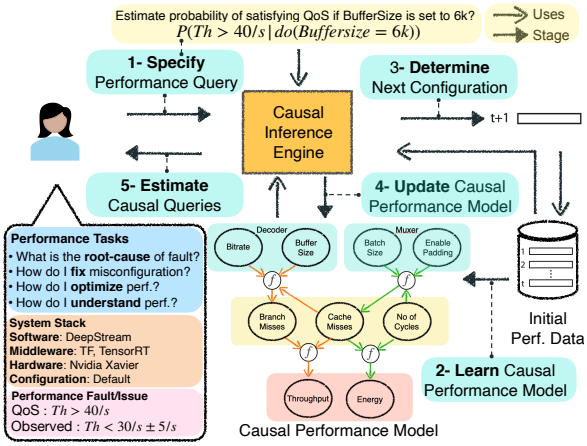

Abbildung 1:

- Performance-Variablen als Knoten für mögliche Konfigurationsparameter (z. B. `Birate`, `Buffer Size` oder `Batch Size`),
- funktionalen Knoten, die funktionale Abhängigkeiten zwischen den Performance-Variablen modellieren,
- kausalen Verbindungen zwischen den Performance-Variablen und den funktionalen Knoten und
- Nebenbedingungen, um notwendige Einschränkungen bei der Modellbildung zu definieren (z. B. dürfen `Cache Misses` nur positive ganzzahlige Werte annehmen).

UNICORN arbeitet in 5 Schritten:

1. Spezifikation der Optimierungsanforderung als Klartext durch den Nutzer
2. Lernen des kausalen Modells mittels vordefinierter Anzahl an Musterkonfigurationen: Hierfür wird der *Fast Causal Inference*-Algorithmus [ref] genutzt, um Kanten von

[1] Unter (System-)Performance werden alle Leistungsmerkmale eines Systems wie beispielsweise Durchsatz und Energieverbrauch zusammengefasst.

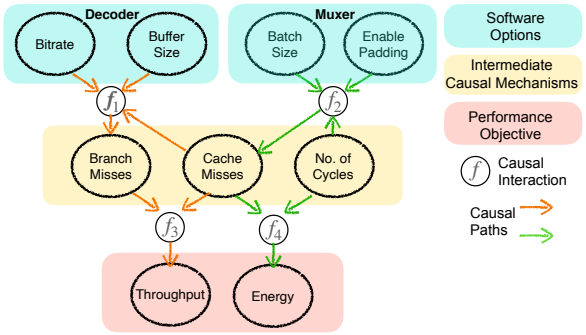

Abbildung 2:

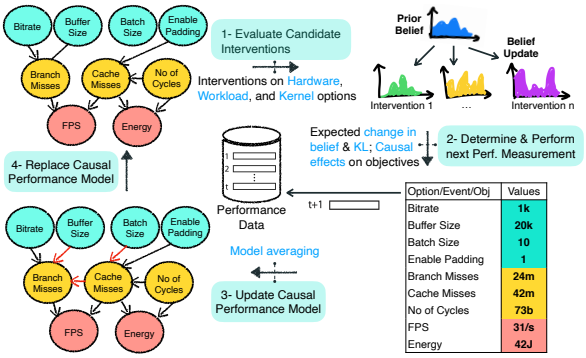

Abbildung 3:

dem zunächst vollständigem probabilistischen Graphen entsprechend der Nebenbedingungen zu entfernen und ungerichtete Kanten entsprechend der kausalen Einflüsse der Performance-Variablen auszurichten.

3. Bestimmung der Folgekonfiguration und Messen der Systemperformance: Für die Folgekonfiguration werden diejenigen Performance-Variablen angepasst, die kausal am stärksten Zusammenhängen, um den Lernprozess zu beschleunigen (*Active Learning*).
4. Inkrementelles Update des kausalen Modells
5. Wiederholung der Schritte 3 und 4 bis ein vordefiniertes Limit (z. B. Laufzeit) überschritten wurde und anschließend Ausgabe

3 Fallbeispiele

In dieser Ausarbeitung werden die Ergebnisse der Evaluation von UNICORN anhand der Fallbeispiele aus dem Originalpaper repliziert.

UNICORN kann im Online- oder Offline-Modus betrieben werden. Im Online-Modus werden die Performance-Metriken direkt auf der zugrundeliegenden Hardware ausgeführt. Der Offline-Modus erlaubt die Reproduktion auf beliebiger Hardware. Da uns die im Paper genutzte Hardware nicht zur Verfügung steht, werden lediglich die Ergebnisse der Offline-Evaluation repliziert.

Die Autoren stellen den Quellcode als Python-Bibliothek auf GitHub zur Verfügung.[2] Eine Dokumentation sowie eine ausführliche Anweisungen zur Replikation der Ergebnisse sind im Repository angehängt. Entsprechend den Anweisungen wurde Docker in Version `20.10.16` auf den Testsystemen installiert.

Bis auf ein fehlendes Paket, das zur Ausführung auf unseren Systemen notwendig war, konnten die Tests ohne Probleme ausgeführt werden. UNICORN erzeugt nach dem Testläufen mit `matplotlib` Graphen. In manchen Fällen konnten die Graphen nicht gespeichert werden, was jedoch nicht auf die eigentlichen Tests und deren Ergebnisse zurückzuführen ist.

Die Replikation findet auf zwei Testsystemen statt:[3]

- `Windows 10 21H2` mit `Intel Core i7-8700K CPU @ 12x3.50 GHz` und `32GB DDR4 RAM @ 3280Mhz`
- `MacOS Monterey 12.4` mit `Apple M1 CPU @ 4x3.20 GHz` und `16 GB LPDDR-DDR4X RAM @ 4266 Mhz`

Den Docker-Containern werden je 4 CPU-Kerne und 8 GB Arbeitsspeicher zur Verfügung gestellt.

Das System der Autoren im Offline-Mode verwendet als Prozessor `Intel Core i7-8700 CPU @ 12x3.20 GHz` und besitzt `31.2 GB` an Arbeitsspeicher, die zugrundeliegende Hardware ist zu unserem ersten System sehr ähnlich. Als Betriebssystem verwenden die Autoren jedoch `Ubuntu 18.04`.

Wir werden im Folgenden die Resultate der drei Kernbehauptungen des Originalpapers reproduzieren:

- UNICORN kann für das Aufsprüren der Ursachen von nicht-funktionalen Fehlern (Latenz, Energieverbrauch) genutzt werden.
- UNICORN kann als Werzeug bei der Durchführung von Performanceoptimierungen unterstützen.
- UNICORN ist effizient, auch wenn sich das Einsatzenvironment ändert.

[2] https://github.com/softsys4ai/unicorn

[3] Die Testsysteme werden im Folgenden kurz als `Win` und `Mac` bezeichnet.

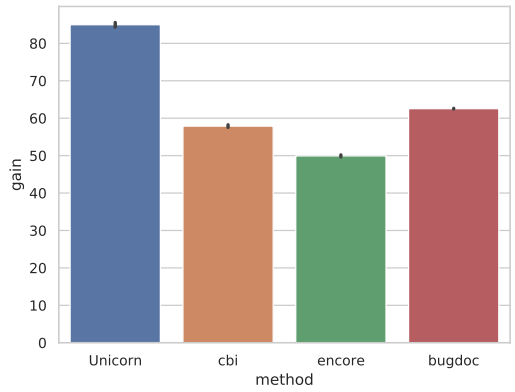

Abbildung 4: Verbesserung des Energieverbrauchs gegenüber Standardkonfiguration bei Einsatz von UNICORN, CBI, ENCORE und BUGDOC. Die Ergebnisse sind für `Win` und `Mac` identisch.

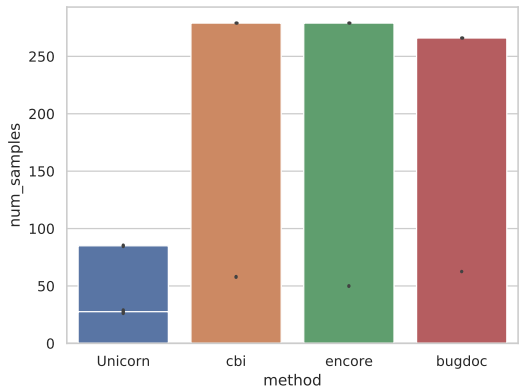

Abbildung 5: Notwendige Anzahl an Samples

4 Replikation der Ergebnisse

Für die Reproduktion der Ergebnisse werden die Anweisungen der Dokumentation in `artifact/REPRODUCE.md` genutzt. Leider konnten bei einigen der Experimente die von UNICORN ausgegeben Graphen nicht gespeichert werden, so dass der zeitliche Verlauf beispielsweise bei der Optimierung nicht ersichtlich ist.

4.1 Erkennung der Ursachen für hohen Energieverbrauch

In diesem Experiment soll die Ursache für den hohen Energieverbrauch ermittelt und eine Lösung hierfür gefunden werden.

Wie in Abbildung 4 zu sehen ist, entspricht der *Gain*, d. h. der Zuwachs an Performance und somit die Verringerung des Energieverbrauchs, bei unseren Experimenten den Ergebnissen der Autoren. Der in Abbildung 6 gezeigte kausale Graph gibt

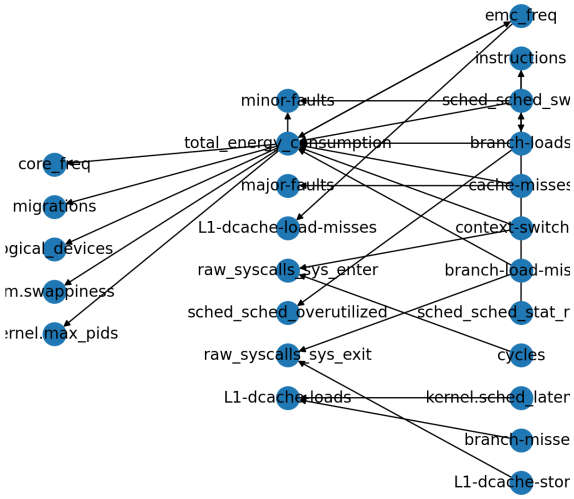

Abbildung 6: Von UNICORN gelernter kausaler Graph des Experiments aus Abschnitt 4.1.

Aufschluss darüber, welche Konfigurationsparameter Einfluss auf den Energieverbrauch haben. Die genauen Werte sind im von UNICORN ausgegeben Log zu sehen:

```
+++++++++++++++++Bug++++++++++++++++++++
memory_growth                          0.5
logical_devices                        1.0
core_freq                        1651200.0
gpu_freq                         1651200.0
emc_freq                       800000000.0
num_cores                              2.0
scheduler.policy                       1.0
vm.swappiness                        100.0
vm.vfs_cache_pressure                500.0
vm.dirty_background_ratio             80.0
vm.drop_caches                         3.0
vm.nr_hugepages                        1.0
vm.overcommit_ratio                   50.0
vm.overcommit_memory                   1.0
vm.overcommit_hugepages                2.0
kernel.sched_child_runs_first          0.0
kernel.sched_rt_runtime_us        500000.0
vm.dirty_bytes                        30.0
vm.dirty_background_bytes             60.0
vm.dirty_ratio                         5.0
swap_memory                            1.0
kernel.max_pids                    32768.0
kernel.sched_latency_ns         24000000.0
kernel.sched_nr_migrate              256.0
kernel.cpu_time_max_percent           50.0
kernel.sched_time_avg_ms            1000.0
Name: 28, dtype: float64
Bug Objective Value 141401
```

```
++++++++++Recommended Fix++++++++++++++
memory_growth                  5.0000e-01
logical_devices                1.0000e+00
core_freq                      1.5744e+06
gpu_freq                       1.6512e+06
emc_freq                       2.1330e+09
num_cores                      2.0000e+00
scheduler.policy               1.0000e+00
vm.swappiness                  1.0000e+02
vm.vfs_cache_pressure          5.0000e+02
vm.dirty_background_ratio      8.0000e+01
vm.drop_caches                 3.0000e+00
vm.nr_hugepages                1.0000e+00
vm.overcommit_ratio            5.0000e+01
vm.overcommit_memory           1.0000e+00
vm.overcommit_hugepages        2.0000e+00
kernel.sched_child_runs_first 0.0000e+00
kernel.sched_rt_runtime_us     5.0000e+05
vm.dirty_bytes                 3.0000e+01
vm.dirty_background_bytes      6.0000e+01
vm.dirty_ratio                 5.0000e+00
swap_memory                    1.0000e+00
kernel.max_pids                6.5536e+04
kernel.sched_latency_ns        2.4000e+07
kernel.sched_nr_migrate        2.5600e+02
kernel.cpu_time_max_percent    5.0000e+01
kernel.sched_time_avg_ms       1.0000e+03
Name: 0, dtype: float64
Unicorn Fix Value 26678
Number of Samples Required 26
```

Wie zu erwarten schlägt UNICORN Änderungen bei den Frequenzen (`core_freq`, `gpu_freq` und `emc_freq`) vor. Die Ergebnisse der Autoren konnten in diesem Fall sehr gut repliziert werden.

4.2 Optimierung der Inferenzzeit

In diesem Experiment wird die minimale Latenz, d. h. die Inferenzzeit, optimiert.

Das Optimum von 12 Sekunden Inferenzzeit erreicht UNICORN bereits nach 69 Iterationen. Auf `Win` benötigt es dafür ca. 280 Minuten, auf `Mac` benötigt es 70 Minuten, die nötige Laufzeit von Unicorn auf `Win` war also sehr viel länger war als auf `Mac` und damit deutlich länger als in der Dokumentation angegeben (ca. 90 Minuten). Grundsätzlich ist hierbei nicht davon auszugehen, dass die zugrundeliegende Hardware dafür verantwortlich ist, da die Autoren bei ihren Experimenten die gleiche CPU verwendet haben. Möglicherweise ist eine Systemkonfiguration außerhalb des Containers, in dem die Berechnungen stattfanden, dafür verantwortlich (Docker von `Win` basiert auf WSL2 was beim Testsystem der Autoren, das auf Ubuntu basiert, nicht der Fall ist).

Auch hier stimmen die Resultate mit denen im Paper überein. UNICORN erreicht das lokale Optimum bereits nach 69 Iterationen und damit schneller als SMAC. Andererseits ist das von UNICORN gefundene Optimum um 8 Sekunden besser als das von SMAC.

4.3 Änderung der Hardware

In diesem Experiment wird die Übertragbarkeit des von UNICORN gelernten Modells untersucht.

The templates include the LaTeX source of this document (`acl.tex`), the LaTeX style file used to format it (`acl.sty`), an ACL bibliography style (`acl_natbib.bst`), an example bibliography (`custom.bib`), and the bibliography for the ACL Anthology (`anthology.bib`).

5 Engines

To produce a PDF file, pdfLaTeX is strongly recommended (over original LaTeX plus dvips+ps2pdf or dvipdf). XeLaTeX also produces PDF files, and is especially suitable for text in non-Latin scripts.

6 Preamble

The first line of the file must be

```
\documentclass[11pt]{article}
```

To load the style file in the review version:

```
\usepackage[review]{acl}
```

For the final version, omit the `review` option:

```
\usepackage{acl}
```

To use Times Roman, put the following in the preamble:

```
\usepackage{times}
```

(Alternatives like txfonts or newtx are also acceptable.)

Please see the LaTeX source of this document for comments on other packages that may be useful.

Set the title and author using `\title` and `\author`. Within the author list, format multiple authors using `\and` and `\And` and `\AND`; please see the LaTeX source for examples.

By default, the box containing the title and author names is set to the minimum of 5 cm. If you need more space, include the following in the preamble:

```
\setlength\titlebox{<dim>}
```

where `<dim>` is replaced with a length. Do not set this length smaller than 5 cm.

7 Document Body

7.1 Footnotes

Footnotes are inserted with the `\footnote` command.[4]

[4]This is a footnote.

Command	Output	Command	Output
`{\"a}`	ä	`{\c c}`	ç
`{\^e}`	ê	`{\u g}`	ğ
`{\`i}`	ì	`{\l}`	ł
`{\.I}`	İ	`{\~n}`	ñ
`{\o}`	ø	`{\H o}`	ő
`{\'u}`	ú	`{\v r}`	ř
`{\aa}`	å	`{\ss}`	ß

Tabelle 1: Example commands for accented characters, to be used in, *e.g.*, BibTeX entries.

7.2 Tables and figures

See Table 1 for an example of a table and its caption. **Do not override the default caption sizes.**

7.3 Hyperlinks

Users of older versions of LaTeX may encounter the following error during compilation:

```
\pdfendlink ended up in
different nesting level
than \pdfstartlink.
```

This happens when pdfLaTeX is used and a citation splits across a page boundary. The best way to fix this is to upgrade LaTeX to 2018-12-01 or later.

7.4 Citations

Table 2 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command `\citet` (cite in text) to get "author (year)" citations, like this citation to a paper by **?**. You can use the command `\citep` (cite in parentheses) to get "(author, year)" citations (**?**). You can use the command `\citealp` (alternative cite without parentheses) to get "author, year" citations, which is useful for using citations within parentheses (e.g. **?**).

7.5 References

The LaTeX and BibTeX style files provided roughly follow the American Psychological Association format. If your own bib file is named `custom.bib`, then placing the following before any appendices in your LaTeX file will generate the references section for you:

```
\bibliographystyle{acl_natbib}
\bibliography{custom}
```

You can obtain the complete ACL Anthology as a BibTeX file from https://aclweb.org/

Output	natbib command	Old ACL-style command
(?)	\citep	\cite
?	\citealp	no equivalent
?	\citet	\newcite
(?)	\citeyearpar	\shortcite

Tabelle 2: Citation commands supported by the style file. The style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.

anthology/anthology.bib.gz. To include both the Anthology and your own .bib file, use the following instead of the above.

```
\bibliographystyle{acl_natbib}
\bibliography{anthology,custom}
```

Please see Section 8 for information on preparing BibTeX files.

7.6 Appendices

Use \appendix before any appendix section to switch the section numbering over to letters. See Appendix A for an example.

8 BibTeX Files

Unicode cannot be used in BibTeX entries, and some ways of typing special characters can disrupt BibTeX's alphabetization. The recommended way of typing special characters is shown in Table 1.

Please ensure that BibTeX records contain DOIs or URLs when possible, and for all the ACL materials that you reference. Use the doi field for DOIs and the url field for URLs. If a BibTeX entry has a URL or DOI field, the paper title in the references section will appear as a hyperlink to the paper, using the hyperref LaTeX package.

Acknowledgements

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTeX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

A Example Appendix

This is an appendix.