

Effiziente Speicherung von Zeitreihen mit Betriebsdaten aus Software-Systemen zur Analyse von Laufzeitanomalien

Florian Lautenschlager



Effiziente Speicherung von Zeitreihen mit Betriebsdaten aus Software-Systemen zur Analyse von Laufzeitanomalien

Der Technischen Fakultät

der Friedrich-Alexander-Universität Erlangen-Nürnberg

zur

Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von

Florian Lautenschlager

aus München

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung: **07.05.2019**

Vorsitzender des Promotionsorgans: **Prof. Dr.-Ing. Reinhard Lerch**

Gutachter: **Prof. Dr. rer. nat. Michael Philippsen**
Prof. Dr.-Ing. Klaus Meyer-Wegener

Berichte aus der Informatik

Florian Lautenschlager

**Effiziente Speicherung von Zeitreihen
mit Betriebsdaten aus Software-Systemen
zur Analyse von Laufzeitanomalien**

D 29 (Diss. Universität Erlangen-Nürnberg)

Shaker Verlag
Düren 2019

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: Erlangen-Nürnberg, Univ., Diss., 2019

Copyright Shaker Verlag 2019

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8440-6785-9

ISSN 0945-0807

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Eidesstattliche Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Ganzes oder als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet. Die Promotionsprüfung in dem angestrebten Doktorgrad wurde nicht anderweitig endgültig nicht bestanden.

Der Friedrich-Alexander-Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl Informatik 2 (Programmiersysteme), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Dissertation einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt, soweit diese nicht bereits abgetreten wurden. Die Dissertation darf elektronisch gespeichert und zum Zwecke der Zitatkontrolle genutzt werden.

Mir ist bekannt, dass der Doktorgrad erst nach Aushändigung der Urkunde geführt werden darf und die erworbenen Rechte erlöschen, wenn die Pflichtexemplare nicht rechtzeitig eingereicht werden.

Florian Lautenschlager

Abstract

When business-relevant software systems work inefficiently or fail, they can lead to economic damage and a loss of reputation. These software systems must therefore meet high stability requirements. If they do not meet these requirements, they can show conspicuous runtime behavior, such as abnormal resource usage, sporadic errors due to synchronization conflicts, or suspicious security-related actions. The analysis of the runtime behavior is a mean of understanding the behavior of the software system. It requires a structured approach as well as a tool chain.

The tool chain consists of tools for collecting, storing and analyzing operational data. Monitoring tools collect any kind of operational data, such as metrics, events or traces. Analyses use statistics, machine learning, etc. to detect abnormalities in the operational data. But there is a gap between the collection and the analysis, since standard time series databases, when used to store operational data, do not optimize for analyzing the runtime behavior. The reason for this is that they use a restricted data model with primitive types and functions for these types, which means that an unforeseen combination of any operational data in analyses is not possible. They also have insufficient means to process asynchronous time series that arise due to the problematic nature of the data collection, e.g., noise or different collection strategies. Finally, they achieve a high storage efficiency, but could be more efficient if they exploit domain-specific characteristics. In summary, these limitations of standard time series databases lead to restrictions when they are used to analyze the runtime behavior of a software-system.

The aim of this thesis is to avoid the limitations of standard time series databases and thus to enable an efficient analysis of the runtime behavior. The thesis provides two contributions: an approach for synchronizing asynchronous time series and the domain-specific time series database *Chronix*. Asynchronous time-discrete time series are converted into time-continuous time series and then are synchronized so that they can be processed as synchronous time-discrete time series. Chronix is optimized for analyzing the runtime behavior. It has a generic multidimensional data model, a plugin mechanism for time series types and functions, respects the approach for synchronizing time series and has a functional-lossless long-term storage. In the quantitative evaluation, Chronix outperforms established time series databases: Not only does Chronix have a lower memory footprint, but it also saves 20%–68% of the hard disk space, is 80%–92% faster to access data, and saves 73%–97% of the runtimes on database analysis functions. The qualitative evaluation also shows the advantages of Chronix based on three case studies from the industry.

Zusammenfassung

Arbeiten geschäftsrelevante Software-Systeme ineffizient oder fallen aus, können sie zu wirtschaftlichem Schaden und einem Imageverlust führen. Diese Software-Systeme müssen daher hohen Stabilitätsansprüchen genügen. Erreichen sie die Ansprüche nicht, können sie auffälliges Laufzeitverhalten zeigen, wie anormale Ressourcennutzung, sporadische Fehler durch Synchronisationskonflikte oder auffällige sicherheitsrelevante Aktionen. Die Analyse des Laufzeitverhaltens ist ein Mittel, um das Verhalten des Software-Systems zu verstehen. Sie erfordert ein strukturiertes Vorgehen und eine Werkzeugkette.

Die Werkzeugkette besteht aus Werkzeugen zum Aufzeichnen, Speichern und Analysieren: Monitoring-Werkzeuge sammeln beliebige Betriebsdaten, wie Metriken, Ereignisse oder Spuren. In Analysen werden Statistik, Machine Learning etc. zum Erkennen von Auffälligkeiten in den Betriebsdaten genutzt. Dazwischen existiert eine Lücke, da Standard-Zeitreibendatenbanken, wenn sie zum Speichern der Betriebsdaten verwendet werden, diese nicht für die Analyse des Laufzeitverhaltens optimiert speichern. Der Grund ist, dass sie ein eingeschränktes Datenmodell mit primitiven Datentypen und Funktionen nutzen, das eine Kombination beliebiger Betriebsdaten in den Analysen nicht erlaubt. Zudem haben sie unzureichende Mittel, um asynchrone Zeitreihen zu verarbeiten, die wegen der Erhebungsproblematik entstehen (Rauschen, unterschiedliche Erhebungsstrategien etc.). Sie erreichen schließlich eine hohe Effizienz beim Speichern, die aber durch das Ausnutzen domänenspezifischer Eigenschaften noch höher sein könnte. Deshalb können Standard-Zeitreibendatenbanken nur begrenzt für die Analyse des Laufzeitverhaltens von Software-Systemen eingesetzt werden.

Die vorliegende Arbeit hat das Ziel, die Einschränkungen der Standard-Zeitreibendatenbanken zu vermeiden und so eine effiziente Analyse des Laufzeitverhaltens zu ermöglichen. Dazu liefert die Arbeit zwei Beiträge: einen Ansatz zum Synchronisieren asynchroner Zeitreihen und die domänenspezifische Zeitreibendatenbank *Chronix*. Asynchrone zeitdiskrete Zeitreihen werden in zeitkontinuierliche Zeitreihen überführt, danach synchronisiert, sodass sie anschließend als synchrone zeitdiskrete Zeitreihen verarbeitet werden können. Chronix ist für die Analyse des Laufzeitverhaltens optimiert: Es hat ein generisches multidimensionales Datenmodell, einen Erweiterungsmechanismus für Zeitreihentypen und Zeitreihenfunktionen, berücksichtigt den Ansatz zum Synchronisieren von Zeitreihen und hat einen effizienten funktional-verlustfreien Langzeitspeicher. In der quantitativen Evaluation übertrifft Chronix etablierte Zeitreibendatenbanken: Chronix hat nicht nur einen geringeren Arbeitsspeicherbedarf, spart 20 %–68 % des Festplattenspeicherbedarfs, ist 80 %–92 % schneller beim Datenzugriff und spart 73 %–97 % der Laufzeiten bei datenbankseitigen Analysefunktionen. Die qualitative Evaluation zeigt die Vorteile von Chronix anhand von drei Fallstudien aus der Industrie.

Danksagung

Danken möchte ich an dieser Stelle zuerst meinem Doktorvater, Herrn Prof. Dr. Michael Philippsen, der mich auf meinem Weg zur Promotion stets durch konstruktive und zielführende Anregungen fachlich und persönlich begleitet und unterstützt hat.

Mein herzlicher Dank gilt zudem Herrn Prof. Dr. Klaus Meyer-Wegener, der ohne Zögern und mit großem Interesse bereit war, die Rolle des Zweitgutachters zu übernehmen.

Außerdem möchte ich mich bei der QAware bedanken, insbesondere bei Dr. Josef Adersberger und Johannes Weigend sowie der ganzen Geschäftsführung, die mein Promotionsvorhaben stets wohlwollend begleiteten.

Ein großer Dank geht auch an Diejenigen, die mir wichtige Impulse für meine wissenschaftliche Arbeit gaben: Tilman Seifert, Prof. Dr. Johannes Siedersleben, Felix Lautenschlager, Prof. Florian Künzner und Andreas Kumlehn.

Bei Julia Meier und Claudia Diepold bedanke ich mich sehr für die stilistische und rechtsschriftliche Überarbeitung dieser Arbeit.

Anna, es ist vollbracht!

Veröffentlichungen

Teile der vorliegenden Arbeit sind bereits in den nachfolgenden Artikeln veröffentlicht. Die Inhalte fließen insbesondere in die Kapitel 3, 5 und 6 ein.

- Lautenschlager, F.; Philippsen, M.; Kumlehn, A.; Adersberger, J.: Chronix: Long Term Storage and Retrieval Technology for Anomaly Detection in Operational Data. In: Proceedings of the USENIX Conference on File and Storage Technologies, Santa Clara, CA., USA, 2017, S. 229–242.
- Lautenschlager, F.; Kumlehn, A.; Adersberger, J.; Philippsen, M.: Fast and efficient operational time series storage: The missing link in dynamic software analysis. In: Proceedings of the Symposium on Software Performance, München, Deutschland, 2015, Softwaretechnik-Trends, Bd. 35, Nr. 3.
- Lautenschlager, F.; Kumlehn, A.; Adersberger, J.; Philippsen, M.: Rahmenwerk zur Ausreißerererkennung in Zeitreihen von Software-Laufzeitdaten. In: Tagungsband Software Engineering & Management, Dresden, Deutschland 2015, S. 177–182.

Die Autoren Florian Lautenschlager, Michael Philippsen, Andreas Kumlehn und Josef Adersberger haben an den drei Artikeln folgende Anteile:

Forschungsfrage: Den Rahmen der Artikel bildet das Forschungsprojekt *Design for Diagnosability* [66]. Es ist ein Kooperationsprojekt zwischen der Friedrich-Alexander-Universität Erlangen-Nürnberg und der QAware GmbH. Florian Lautenschlager, Michael Philippsen und Josef Adersberger haben den Förderantrag formuliert. Florian Lautenschlager hat die Forschungsfragen der Artikel herausgearbeitet.

Analyse, Implementierung, Evaluation, Artikel: Florian Lautenschlager hat die Analyse der Forschungsfrage, die Implementierung des Ansatzes sowie die Evaluation durchgeführt und die initialen Fassungen der Artikel formuliert.

Darstellung, Klarheit, Ergänzungen: Michael Philippsen redigierte als kritischer Gutachter die initialen Fassungen. Er lieferte Beiträge zur Darstellung, Klarheit sowie zur Nachvollziehbarkeit der Inhalte und regte ergänzende Auseinandersetzungen an. Andreas Kumlehn und Josef Adersberger trugen als Teil des Projektteams zur inhaltlichen Diskussion bei und initiierten sprachliche und stilistische Korrekturen.

Inhaltsverzeichnis

Abstract	i
Zusammenfassung	iii
Danksagung	v
Veröffentlichungen	vii
1 Einleitung	1
1.1 Motivation	2
1.2 Zielsetzung und Beitrag	4
1.3 Aufbau und Inhalt	6
2 Analyse des Laufzeitverhaltens von Software-Systemen	9
2.1 Begriffe	10
2.1.1 Software-Komponente, -Anwendung, -System	10
2.1.2 Betriebsdaten	11
2.1.3 Laufzeitverhalten	11
2.1.4 Laufzeitanomalie	11
2.1.5 Laufzeit-Antimuster	11
2.1.6 Zeitreihe	12
2.1.7 Zeitreihendatenbank	12
2.1.8 Statische und dynamische Analyse	14
2.1.9 Instrumentierung und Messfühler	14
2.1.10 Mehraufwand	14
2.1.11 Monitoring, Profiling, Tracing und Debugging	15
2.1.12 Software Performance Engineering	16
2.2 Festlegen des Analyserahmens	16
2.2.1 Monitoring-Modell	17
2.2.2 Beeinflussende Faktoren des Laufzeitverhaltens	19
2.2.3 Monitoring-Kategorien	20
2.2.4 Anforderungen an eine domänenspezifische Speichertechnologie	24
2.3 Erheben von Betriebsdaten	25
2.3.1 Grundlage der Erhebung	25
2.3.2 Erhebungsstrategien von Software-Monitoren	26
2.3.3 Kategorien von Betriebsdaten	27
2.3.4 Anforderungen an eine domänenspezifische Speichertechnologie	28

2.4	Abbilden des Laufzeitverhaltens auf Zeitreihen mit Betriebsdaten	28
2.4.1	Repräsentieren des Systemzustands	29
2.4.2	Repräsentieren des Laufzeitverhaltens	29
2.4.3	Darstellen des Laufzeitverhaltens mittels Zeitreihen	30
2.4.4	Anforderungen an eine domänenspezifische Speichertechnologie . .	31
2.5	Erkennen von Anomalien im Laufzeitverhalten	32
2.5.1	Einflussfaktoren bei der Wahl der Anomalieerkennung	32
2.5.2	Zeitreihen, Anomalien und Betrachtungspunkte	33
2.5.3	Analyseverfahren für Metriken, Ereignisse und Spuren	34
2.5.4	Anforderungen an eine domänenspezifische Speichertechnologie . .	39
2.6	Werkzeugkette zur Analyse des Laufzeitverhaltens	39
2.7	Anforderungen an eine domänenspezifische Speichertechnologie	43
2.7.1	Generisches multidimensionales Datenmodell	43
2.7.2	Analysemöglichkeiten und Funktionserweiterungen	44
2.7.3	Synchronisieren von Zeitreihen	44
2.7.4	Effizienter funktional-verlustfreier Langzeitspeicher	44
3	Verwandte Arbeiten	47
3.1	Generisches multidimensionales Datenmodell	48
3.2	Analysemöglichkeiten und Funktionserweiterungen	51
3.3	Synchronisieren von Zeitreihen	54
3.4	Effizienter funktional-verlustfreier Langzeitspeicher	58
3.5	Zusammenfassung	63
4	Synchronisieren von Zeitreihen und Funktionsdefinitionen	65
4.1	Verwandte Arbeiten	66
4.2	Betrachtungsvarianten von Zeitreihen	66
4.3	Definitionen einer Zeitreihe	68
4.3.1	Zeitdiskrete Zeitreihe	68
4.3.2	Zeitkontinuierliche Zeitreihe	71
4.4	Abbildung zeitdiskreter auf zeitkontinuierliche Zeitreihen	72
4.5	Synchronisieren asynchroner zeitdiskreter Zeitreihen	75
4.6	Operationen und Aktionen auf zeitdiskreten Zeitreihen	78
4.6.1	Interpretation unbekannter Werte	79
4.6.2	Funktionsparameter für Operationen und Aktionen	79
4.6.3	Unäre und binäre Operationen	79
4.6.4	Unäre und binäre Aktionen	81
4.6.5	Verkettung von Operationen und Aktionen	82
4.6.6	Verarbeitung von Mengen aus zeitdiskreten Zeitreihen	84
4.7	Univariate und multivariate Zeitreihen	85
4.8	Zusammenfassung	85

5	Zeitreihendatenbank zur Analyse des Laufzeitverhaltens	87
5.1	Datenbankentwurf	88
5.1.1	Generische und leseoptimierte Speicherung von Zeitreihen	88
5.1.2	Konzeptuelle Ebene	90
5.1.3	Logische Ebene	93
5.1.4	Physische Ebene	95
5.2	Anfragesprache und Funktionsausführung	100
5.2.1	Syntax der Chronix Anfragesprache	100
5.2.2	Funktionsausführung	103
5.2.3	Funktionskontext	104
5.3	Erweiterungsmechanismus für Zeitreihen und Funktionen	105
5.3.1	Typdefinition	106
5.3.2	Zeitreihendefinition	108
5.3.3	Zeitreihenfunktionsdefinition	110
5.3.4	Chronix Client: Nutzerseitige Verwendung von Chronix	112
5.3.5	Technisches Rahmenwerk zum Laden der Definitionen	114
5.4	Effizienter funktional-verlustfreier Langzeitspeicher	116
5.4.1	Optionale Transformation	117
5.4.2	Attribute und Bündel	118
5.4.3	Kompression	120
5.4.4	Multidimensionaler Speicher	123
5.5	Einstellen der Architekturparameter	126
5.5.1	Projekte, Betriebsdaten und Infrastruktur	127
5.5.2	DDC-Grenzwert	129
5.5.3	Kompressionsparameter	130
5.6	Zusammenfassung	134
6	Evaluation	137
6.1	Ziele der Evaluation	137
6.2	Messaufbau	139
6.2.1	Realisierung des Messaufbaus	140
6.2.2	Projekte, Betriebsdaten und Analysen	143
6.3	Quantitative Evaluation	146
6.3.1	Evaluierte Standard-Zeitreihendatenbanken	147
6.3.2	Arbeitsspeicherbedarf	148
6.3.3	Festplattenspeicherbedarf	149
6.3.4	Datenzugriffszeiten	150
6.3.5	Analysezeiten	151
6.3.6	Standardeinstellungen für Chronix	154
6.3.7	Zusammenfassung	154
6.4	Qualitative Evaluation	155
6.4.1	Steigende Dateireferenzen	155
6.4.2	Regressionsanalyse von Anwendungsversionen	159

6.4.3	Chronix als Langzeitspeicher von Prometheus	163
6.4.4	Zusammenfassung	167
7	Abschluss	169
7.1	Fazit	169
7.2	Ausblick	172
7.3	Chronix in anderen Projekten	173
	Abbildungsverzeichnis	176
	Tabellenverzeichnis	177
	Listingverzeichnis	179
	Literaturverzeichnis	179
	Abkürzungsverzeichnis	201
	Glossar	203