Dissertationen

Matthias Rohr: Workload-sensitive Timing Behavior Analysis for Fault Localization in Software Systems

1. Gutachter: Prof. Dr. Wilhelm Hasselbring (Universität Kiel)
2. Gutachter: Prof. Dr. Lars Grunske (Universität Stuttgart)
Datum der Prüfung: 02.10.2014

Zusammenfassung:
Die Analyse von Zeitverhalten wie z.B. Antwortzeiten von Software-Operationen ist oft schwierig wegen der hohen statistischen Varianz. Diese Varianz gefährdet sogar die Anwendbarkeit von statistischen Verfahren. In dieser Arbeit wird eine Methode zur Verbesserung der Analyse von Antwortzeiten mit hoher statistischer Varianz vorgestellt.

Der vorgestellte Ansatz ist in der Lage, einen Teil der Varianz aus dem gemessenen Zeitverhalten anhand von Aufrufsequenzen und Schwankungen in der Nutzungsintensität zu erklären. Dadurch kann praktisch Varianz aus den Messdaten entfernt werden, was die Anwendbarkeit von statistischen Analysen in Bezug auf Verlässlichkeit, Präzision und Geschwindigkeit (z.B. kürzere Messperiode und Simulationsdauer) verbessern kann. Der Hauptbeitrag dieser Arbeit liegt in den zwei Verfahren TracSTA (Trace-Context-Sensitive Timing Behavior Analysis) und WiSTA (Workload-Intensity-Sensitive Timing Behavior Analysis). TracSTA verwendet die Form des Aufrufflusses (d.h. die Form der Aufrufsequenz, in die ein Methodenaufruf eingebettet ist). WiSTA wertet die Nutzungsintensität aus (z.B. Anzahl gleichzeitig ausgeführter Methoden). Dies resultiert in kontextspezifischen Antwortzeitprofilen.

In mehreren Fall- und Feldstudien wird die Anwendbarkeit und die Wirksamkeit evaluiert. Es zeigt sich ein deutlicher Zusammenhang zwischen dem Zeitverhalten und den von TracSTA und WiSTA betrachteten Einflussfaktoren. Zusätzlich wird als Anwendungsszenario ein Ansatz zur Fehlerlokalisierung vorgestellt, welcher von TracSTA und WiSTA bereitgestellte Antwortzeiten zur Anomalieerkennung verwendet.

Veröffentlicht als: Matthias Rohr: Workload-sensitive Timing Behavior Analysis for Fault Localization in Software Systems, 2015.
Online unter http://eprints.uni-kiel.de/27337/

André van Hoorn: Model-Driven Online Capacity Management for Component-Based Software Systems

André van Hoorn: Model-Driven Online Capacity Management for Component-Based Software Systems

1. Gutachter: Prof. Dr. Wilhelm Hasselbring (Universität Kiel)
2. Gutachter: Prof. Dr. Ralf Reussner (Karlsruhe Institute of Technology)
3. Gutachter: Prof. Dr. Samuel Kounev (Universität Würzburg)
Datum der Prüfung: 18.09.2014

Zusammenfassung:
Kapazitätsmanagement ist eine Kernaktivität beim Entwurf und beim Betrieb von verteilten Softwaresystemen. Es umfasst die Bereitstellung von Rechenzentrumsressourcen und die Verteilung der Softwarekomponenten auf diese Ressourcen. Ziel ist die kontinuierliche Bereitstellung einer angemessenen Kapazität, so dass Dienstgütevereinbarungen (SLAs) eingehalten werden, während Investitions- und Betriebskosten möglichst niedrig sind. Traditionelle Kapazitätsmanagementstrategien sind eher statisch und pessimistisch, d.h. Ressourcen werden für Lastspitzen ausgelegt. Insbesondere betriebliche Informationssysteme sind jedoch starken Lastschwankungen ausgesetzt. Hier führt das eben genannte statische Kapazitätsmanagement zu einer niedrigen Ressourceneffizienz und damit zu unnötig hohen Gesamtkosten.

Im Laufe der vergangenen Jahre wurden Technologien für dynamische Rechenzentrumsinfrastrukturen verfügbar, z.B. durch Cloud-Computing-Produkte. Diese Technologien stellen die Basis für dynamisches Kapazitätsmanagement zur Laufzeit, d.h. die kurzfristige Anpassung der bereitgestellten Kapazität auf Basis des aktuellen Bedarfs. Weil manuelles Kapazitätsmanagement nicht

praktikabel ist, wurden automatische Ansätze vorgestellt. Allerdings sehen diese Ansätze grobgranulare Adaptations-Aktionen und -Entscheidungen vor, die auf aggregierten Messungen auf Systemebene basieren.

Diese Dissertation schlägt einen modellgetriebenen Ansatz, namens SLAstic, zum Kapazitätsmanagement von verteilten, komponentenbasierten Softwaresystemen zur Laufzeit vor. Die Kernbeiträge dieses Ansatzes sind a) Modellierungssprachen zur Repräsentation von Architekturinformationen des kontrollierten Softwaresystems, b) ein architekturbasiertes Laufzeit-Kapazitätsmanagement-Rahmenwerk basierend auf der MAPE-K-Architektur, c) modellgetriebene Techniken zur Automatisierung des Ansatzes, d) architektonische Laufzeit-Rekonfigurationsoperationen zur Steuerung der Systemkapazität, e) sowie eine Integration des Palladio-Komponentenmodells. Eine qualitative und quantitative Bewertung des Ansatzes erfolgt durch Kombination von Simulation, Laborexperimenten und Fallstudien.

Veröffentlicht als: André van Hoorn: Model-Driven Online Capacity Management for Component-Based Software Systems, 2014.
Online unter http://eprints.uni-kiel.de/25969/

Jan Waller: Performance Benchmarking of Application Monitoring Frameworks

1. Gutachter: Prof. Dr. Wilhelm Hasselbring (Universität Kiel)
2. Gutachter: Prof. Dr. Klaus Schmid (Universität Hildesheim)
Datum der Prüfung: 12.12.2014

Zusammenfassung:
Die Überwachung (Monitoring) von kontinuierlich arbeitenden Software-Systemen auf der Applikationsebene gibt Einblicke in ihr dynamisches Verhalten und kann dabei helfen, ihre Anforderungen an Performance und Verfügbarkeit während der Laufzeit einzuhalten. Abhängig von der Anzahl und Lage der verwendeten Monitoring-Sonden, kann eine solche Überwachung zu erheblichen zusätzlichen Laufzeitkosten (Overhead) auf dem überwachten System führen. Um die Instrumentierung eines Systems zu verbessern und den durch die Überwachung verursachten Overhead zu reduzieren, ist es notwendig, die Kosten jeder Sonde genau zu kennen.

Während viele Monitoring-Frameworks behaupten, minimale Auswirkungen auf die Performance zu haben, werden diese Ansprüche oft nicht mit einer detaillierten Untersuchung der tatsächlichen Überwachungskosten untermauert. Benchmarks stellen eine effektive und kostengünstige Möglichkeit für diese Untersuchungen dar. Jedoch gibt es derzeit keine Benchmarks, die speziell auf den Overhead von Monitoring abzielen. Außerdem gibt es keine etablierte Benchmark-Engineering-Methodik, die Richtlinien für die Entwicklung, Durchführung und Analyse von Benchmarks festlegt.

Diese Arbeit stellt einen Benchmark-Ansatz vor, der den Overhead von Monitoring-Frameworks misst. Die Hauptbeiträge dieses Ansatzes sind: 1) eine Definition der häufigsten Ursachen von Monitoring-Overhead, 2) eine allgemeine Benchmark-Engineering-Methodik, 3) der MooBench Mikro-Benchmark, um die verschiedenen Ursachen von Monitoring-Overhead zu messen und zu quantifizieren und 4) detaillierte Performance-Untersuchungen von drei verschiedenen Monitoring-Frameworks auf der Applikationsebene. Ausführliche Experimente zeigen die Machbarkeit und Praktikabilität des Ansatzes und validieren die Benchmark-Ergebnisse. Das entwickelte Benchmark-Tool wird als Open-Source-Software zur Verfügung gestellt und die Ergebnisse aller Experimente stehen zum Download bereit, um die weitere Validierung und Replikation der Ergebnisse zu ermöglichen.

Veröffentlicht als: Jan Waller: Performance Benchmarking of Application Monitoring Frameworks, 2014.
Online unter http://eprints.uni-kiel.de/26979/