KAPITEL 1

EINLEITUNG

In komplexen Softwaresystemen erlaubt die Aufzeichnung von auftretenden Ereignissen Rückschlüsse auf Vorgänge und Zusammenhänge innerhalb des beobachteten Systems. Grundprobleme sind dabei die bei der Aufzeichnung entstehende große Datenmenge und die starke Systembeeinflussung. Das Ziel der in dieser Arbeit vorgestellten Ansätze ist es, zur Ausführungszeit eines Softwaresystems eine effiziente Aufzeichnung und Analyse von Ereignissen zu ermöglichen.

Im weiteren Verlauf dieses Kapitels werden der Kontext der Arbeit und die betrachtete Problemstellung dargestellt. Aufbau und Struktur der Arbeit sowie die Beiträge der Arbeit werden beschrieben. Weiterhin sind grundsätzliche Anmerkungen zu Sprache und Notation der Arbeit enthalten.

1.1 Kontext

Die Beobachtung von komplexen Softwaresystemen durch Aufzeichnung von auftretenden Ereignissen im System (*event tracing*) wird in unterschiedlichen Bereichen eingesetzt:

Systemanalyse: Die Untersuchung aufgezeichneter Ereignisfolgen kann zu einem verbesserten Verständnis von Vorgängen in einem System führen, beispielsweise von Vorgängen im Betriebssystemkern [48] oder in einem komplexeren Gesamtsystem [27]. Ganz allgemein können aus dem beobachteten Verhalten Rückschlüsse auf beteiligte Komponenten und Zusammenhänge im System gezogen werden.

Fehleranalyse: Einige Problemklassen, beispielsweise Performanzprobleme, lassen sich nur zur Laufzeit des betroffenen Systems analysieren. Die Aufzeichnung von Interaktionen zwischen beteiligten Systemkomponenten [19, 38] kann die Fehlersuche unterstützen. Die Analyse soll also Erkenntnisse über die Ursache von Problemen erbringen.

Überwachung zur Laufzeit: Die kontinuierliche Aufzeichnung von Ereignissen während der Systemlaufzeit kann angewandt werden, um beispielsweise die Einhaltung von gewünschtem Antwortzeitverhalten [21] zu überwachen oder um Einbrüche in das System festzustellen [66].

Bei den vorgestellten Ansätzen wird eine konkrete Implementierung eines Systems instrumentiert, das heißt die aufzuzeichnenden Ereignisse werden ausgewählt. Anschließend erfolgt die Systembeobachtung - eine reguläre oder leicht modifizierte Anwendung wird ausgeführt und die angezeigten Ereignisse werden gespeichert. Basierend auf den gespeicherten Ereignissen

werden Rückschlüsse auf das Systemverhalten gezogen und gegebenenfalls die Anwendung aufgrund der Beobachtungsergebnisse angepasst. Dabei steht - im Gegensatz zur Modellprüfung oder der Systemverifikation - die Analyse des tatsächlichen Verhaltens einer vorliegenden Systemimplementierung im Vordergrund und nicht der Vergleich mit einem Modell oder eine Spezifikation.

Je nach Art der durchzuführenden Analyse kann die Auswertung der aufgezeichneten Ereignisse zeitaufwändig und komplex sein. Weitere Einschränkungen ergeben sich aus der Systeminstrumentierung und der zu bewältigenden Datenmenge: das Systemverhalten wird durch die Instrumentierung beeinflusst und zahlreiche auftretende Ereignisse müssen zuverlässig, das heißt möglichst vollständig, aufgezeichnet werden.

Zur Überwachung und Analyse stehen verschiedene Werkzeug-Typen zur Verfügung. Ein Überblick wird in Kapitel 2 gegeben. In der vorliegenden Arbeit werden Werkzeuge zur *dynamischen Analyse* von Systemen betrachtet. Mit statischer Analyse werden Systeme untersucht, die sich nicht in Ausführung befinden - beispielsweise basierend auf dem Quelltext einer Anwendung. Die dynamische Analyse erlaubt die Untersuchung vom Laufzeitverhalten eines Systems. Werkzeuge zur dynamischen Analyse können in *sampling-basiert* und *tracing-basiert* eingeteilt werden. Sampling-basierte Werkzeuge bestimmen den Zustand des analysierten Systems periodisch und analysieren das Verhalten des Systems mit Methoden der Statistik. Tracing-basierte Werkzeuge analysieren, wie beschrieben, (vollständige) Ketten von Ereignissen im System.

Beide Werkzeugtypen haben spezifische Vor- und Nachteile: Tracing-basierte Werkzeuge erfordern die Erzeugung von Ereignissen im zu analysierenden System. Der entsprechende Programmcode muss entweder bereits in der Software vorhanden sein (und bei Bedarf aktiviert werden) oder zur Laufzeit eingefügt werden. Weiterhin ist es im Allgemeinen nicht möglich die Anzahl der im Rahmen der Analyse erzeugten Ereignisse vorherzusagen. Demnach ist auch der Einfluss der Analyse auf das beobachtete System nicht vorhersagbar.

Sampling-basierte Werkzeuge erfordern Messpunkte im analysierten System, die periodisch abgefragt werden können. Der Einfluss der Analyse kann durch Einstellen der Periode bestimmt werden. Sampling von Systemzuständen kann jedoch Änderungen im System nicht erfassen, die zwischen zwei Messzeitpunkten auftreten. Weiterhin gilt, dass manche Vorgänge im System sich nicht durch eine Folge von Messwerten beschreiben oder erklären lassen.

Nach abgeschlossener Analyse wird gegebenenfalls das System verändert, um auf identifizierte Probleme zu reagieren. Entsprechende Systemänderungen sind beispielsweise eine Rekonfiguration der Systembestandteile, das Einspielen von Patches oder der Neustart von Systemteilen.

In der Regel erfolgen die Analyse und die resultierende Anpassung des Systems zeitlich getrennt und werden, beispielsweise in den Bereichen der System- und Fehleranalyse, manuell und nur bei Bedarf ausgeführt.

In der vorliegenden Arbeit werden tracing-basierte Konzepte zur Systembeobachtung und -anpassung untersucht. Es wird eine generische Architektur entwickelt, die es erlaubt tracing-basierte Systemanalyse in Betriebssystemen und Anwendungen zu verwenden. Die Grundidee ist dabei (Werkzeug-unterstützt oder manuell) Beschreibungen von Ereigniskonstellationen zu spezifizieren, die Reaktionen auslösen. Eine solche Reaktion kann beispielsweise ein Eintrag in einer Logdatei, der Neustart einer Systemkomponente oder eine Rekonfiguration des Systems sein.

1.2 Problemdefinition

Die Verwaltung komplexer Systeme ist eine zeit- und arbeitsintensive Aufgabe - ständige Änderungen der Rahmenbedingungen einer Anwendung erfordern ständige Anpassung der Konfiguration und Ausführung der Systeme. Konzepte zur Optimierung und Automatisierung des Zyklus aus Analyse und Systemanpassung werden daher wichtiger um selbst-adaptive Anwendungen zu implementieren.

Um die Systemverwaltung zu automatisieren, werden heute vorwiegend Regelkreis-basierte Ansätze verwendet: aus gemessenen Werten werden mit Hilfe eines Systemmodells neue Konfigurationsparameter bestimmt, die dann über Aktuatoren gesetzt werden. Regelkreis-basierte Konzepte haben jedoch Nachteile, die sich aus dem *Sampling* des Systemzustandes ergeben: Tritt eine Systemzustandsänderungen zwischen zwei Messzeitpunkten auf, kann diese unter Umständen nicht erfasst werden. Weiterhin lassen sich nicht alle Aspekte der Systemverwaltung auf die Erfassung und Beeinflussung einzelner Messwerte abbilden.

Tracing bietet demgegenüber verschiedene Vorteile: Werden entsprechende Ereignisse generiert, sind Vorgänge wie der Ausfall von Systemkomponenten oder Angriffe einfacher zu erkennen. Auch für die Systemanalyse ist Tracing hilfreich, da Ketten von Ereignissen für das Systemverständnis mehr Informationen enthalten können als die Beobachtung der Entwicklung eines Messwertes im Laufe der Zeit.

Die Verwendung von Tracing zur Überwachung von Systemen zeigt jedoch ebenfalls spezifische Nachteile:

- Im Allgemeinen verursacht Tracing einen hohen Overhead, also zusätzliche Last im System. Bei sehr vielen auftretenden Ereignissen kann das System unter Umständen unbenutzbar werden.
- Die Aufzeichnung von Ereignissen führt (je nach Anzahl und Typ der Ereignisse) zu sehr großen Datenmengen. Diese müssen gespeichert und anschließend analysiert werden.
- Das (automatisierte) Analysieren der Ereignisströme erfordert komplexe Mustererkennung. Je nach konkretem Analyseziel müssen unter Umständen sehr große Datenmengen nach seltenen Ereigniskonstellationen durchsucht werden.
- Viele Werkzeuge führen die Analyse von Ereignisströmen zeitlich getrennt von der Aufzeichnung, also nachträglich, durch. Dies verhindert eine schnelle Reaktion auf erkannte Ereigniskonstellationen.

In der vorliegenden Arbeit sollen diese Tracing-spezifischen Nachteile untersucht und Lösungsmöglichkeiten aufgezeigt werden. Ziel ist es, die Vorteile von Tracing zur automatisierten Systemanalyse verfügbar zu machen.

Dabei wird nicht angestrebt Sampling-basierte Werkzeuge durch Tracing-basierte zu ersetzen. Vielmehr sollen die Tracing-spezifischen Nachteile in Szenarien, in denen der Einsatz Tracing-basierter Werkzeuge sinnvoll ist, verringert werden.

Fragestellungen, die in diesem Zusammenhang untersucht werden, sind beispielsweise: Wie kann die Menge der Daten, die beim Tracing aufgezeichnet werden muss, reduziert werden? Wie kann schnell auf erkannte Muster im Ereignisstrom reagiert werden?

Der Grundgedanke der in dieser Arbeit vorgestellten Ansätze ist dabei, dass fein-granulare Ereignisse im Betriebssystemkern zu komplexeren Ereignissen auf einer höheren Abstraktionsebene zusammengefasst analysiert werden können. Die vorgestellten Konzepte erlauben die Beschreibung und Erkennung von Ereigniskonstellationen, die solche komplexe Ereignisse bilden. Neben der Erkennung dieser Ereigniskonstellationen zur Laufzeit soll auch eine sofortige Reaktion ermöglicht werden. Dies ermöglicht dann Tracing-basierte Ansätze zur automatischen Systemadaption zu verwenden, beispielsweise im Rahmen von selbst-adaptiven Anwendungen.

1.3 Anwendungsszenarien

Der Schwerpunkt der vorliegenden Arbeit liegt auf der Entwicklung und Implementierung einer Laufzeitumgebung zur Verarbeitung von Ereignisströmen im Betriebssystemkern. Die Laufzeitumgebung ermöglicht die sofortige Reaktion auf erkannte Ereigniskonstellationen im Ereignisstrom.

Diese Konzepte bilden die Grundlage für zahlreiche Anwendungmöglichkeiten, von denen einige in diesem Abschnitt kurz dargestellt werden. Ausgewählte Szenarien werden im Rahmen von Machbarkeitsstudien im Kapitel 5 untersucht, andere erfordern über diese Arbeit hinausgehende Forschung.

Softwareprüfstand: Phänomene wie überlastete Synchronisationsobjekte (*lock contention*) oder Prioritäteninvertierung lassen sich meist erst zur Laufzeit des Systems unter realen Lastbedingungen erkennen. Mit einem Prüfstand, der nach allgemein bekannten, ungünstigen Ereigniskonstellationen zur Laufzeit sucht, kann ein gegebenes System bezüglich solcher Probleme analysiert werden.

Überwachung von Anwendungen: Während der Laufzeit eines Systems können verschiedene Ereigniskonstellationen auftreten, die ganz unterschiedliche Aktionen erfordern: Bei einem erkannten Angriff auf das System können Gegenmaßnahmen eingeleitet werden und beispielsweise die verdächtige Aktivität in einen isolierten Bereich verschoben werden. Bei der Anmeldung eines Benutzers und der Ausführung von Funktionen kann beispielsweise geprüft werden, ob der Benutzer diese Funktion tatsächlich ausführen darf oder die Funktionsausführung protokolliert werden.

Selbst-adaptive Systeme: Eine Folge von Ereignissen kann als Basis einer Rekonfiguration des Systems dienen, indem beispielsweise eine als fehlerhaft erkannte Systemkomponente ersetzt wird. Eine weitere Möglichkeit ist die Umschaltung zwischen verschiedenen Implementierungsstrategien, wenn die auftretenden Ereignisse im System ineffizientes Verhalten anzeigen oder sich wichtige Umgebungsbedingungen geändert haben.

Skripting: Durch die sofortige Ausführung von Aktionen nach der Erkennung von bestimmten Ereigniskonstellationen kann das Gesamtsystemverhalten beeinflusst werden. Auf diese Weise können bestimmte Verhaltensweisen erzwungen werden oder nachträglich Funktionalität im System ergänzt werden.

Mit den dargestellten Anwendungsmöglichkeiten leistet die entwickelte Laufzeitumgebung einen Beitrag zur effizienten Verwaltung und Entwicklung komplexer Systeme auf verschiedenen Ebenen - vom Betriebssystemkern bis zur Anwendung selbst.

1.4 Beitrag dieser Arbeit

Im Rahmen dieser Arbeit wurde eine Laufzeitumgebung für die effiziente Verarbeitung von Ereignisströmen auf der Ebene des Betriebssystemkerns entwickelt. Die Laufzeitumgebung ermöglicht dabei:

1. die Erkennung von Konstellationen mehrerer Ereignisse im systemweiten Ereignisstrom
2. die sofortige Verarbeitung und Analyse direkt während des Auftretens der Ereignisse
3. die Ausführung von Aktionen sofort nach der Erkennung bestimmter Ereigniskonstellationen

Die Möglichkeit zur synchronen Reaktion auf erkannte Ereigniskonstellationen schließt die üblicherweise vorhandene Lücke in tracing-basierter Systemüberwachung zwischen Systemanalyse und möglicher Reaktion, also der Anpassung des Systems: die Anpassung erfolgt *nicht* post-mortem, sondern bei weiterhin aktivierter Systemüberwachung. Außerdem kann bei sofortiger Analyse des Ereignisstroms die aufzuzeichnende Datenmenge reduziert werden und auf eine explizite Speicherung aller auftretender Ereignisse verzichtet werden - die Laufzeitumgebung sammelt ausschließlich für die zu erkennenden Ereigniskonstellationen relevante Informationen.

Die Bestandteile der Arbeit sind im Einzelnen:

Instrumentierung: Der *Windows Monitoring Kernel* wird vorgestellt, eine Infrastruktur zur Aufzeichnung und Analyse von Ereignissen im Windows Betriebssystemkern.

Sprachspezifikation: Es wird eine Sprache zur Spezifikation von Regeln vorgestellt, die es ermöglicht Konstellationen von Ereignissen zu beschreiben und mit Aktivitäten zu verknüpfen, die ausgeführt werden sollen, wenn entsprechende Konstellationen in einem Ereignisstrom erkannt werden.

Besonderheiten im Umfeld der Betriebssysteme: Ereignisstromverarbeitung wird bisher vor allem in Geschäftsanwendungen eingesetzt. Die Integration in einen Betriebssystemkern unterscheidet sich davon erheblich. Unterschiede in den Bereichen Ereignisbeschreibung und Implementierungsmöglichkeiten werden identifiziert. Die Besonderheiten werden verwendet, um neue Verarbeitungsmöglichkeiten zu realisieren.

Implementierung: Die Konzepte wurden implementiert; die entstandene Laufzeitumgebung zur Ereignisstromverarbeitung im Betriebssystemkern ermöglicht die effiziente Erkennung von Ereigniskonstellationen auch bei hohen Ereignisraten.

Modellierung und Komplexitätsabschätzung: Mit Hilfe stochastischer Modelle von Ereignisströmen in einem System und absorbierender Markov-Ketten als Modell für die Ereignisstromverarbeitung wurden Eigenschaften der entwickelten Laufzeitumgebung analysiert.

Anwendung: Beispielanwendungen und komplexere Fallstudien werden vorgestellt und damit die praktische Anwendbarkeit der entwickelten Konzepte belegt.

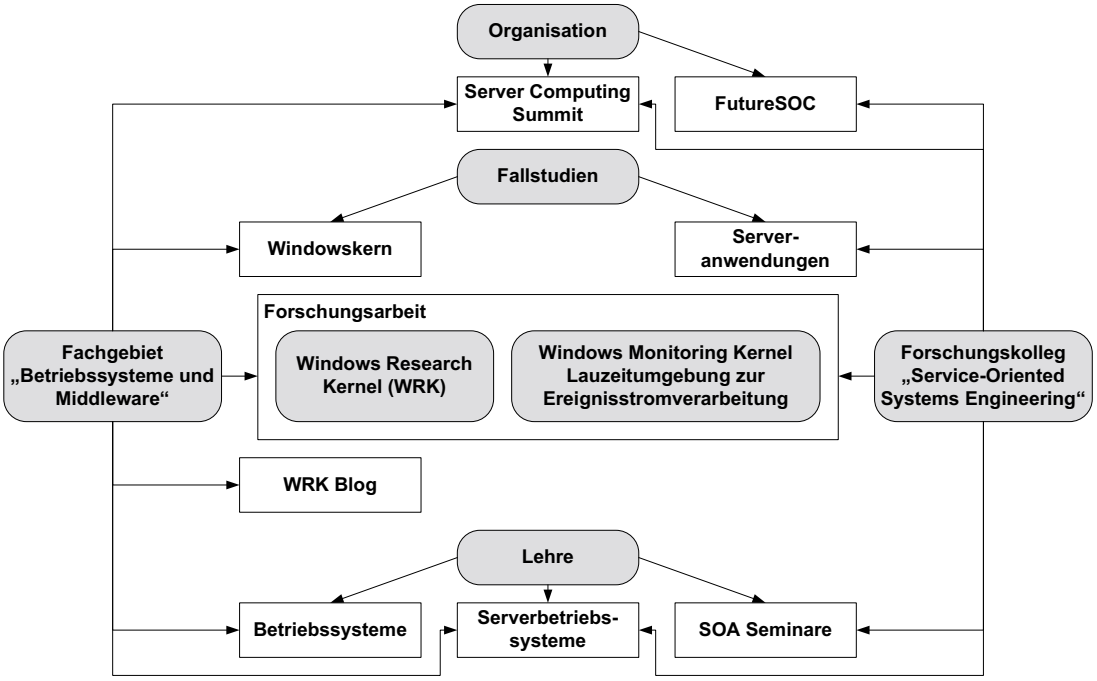

Abbildung 1.1: Umfeld der Arbeit

Die Integration der Konzepte in den Betriebssystemkern erlaubt die direkte Verwendung der entwickelten Konzepte in einer Vielzahl von Anwendungen: die Programmsysteme oberhalb des Betriebssystems müssen nicht oder nur geringfügig angepasst werden.

Die Arbeit wurde im Rahmen des HPI Forschungskollegs „Service-Oriented Systems Engineering“ am Fachbereich für Betriebssysteme und Middleware angefertigt. Dementsprechend war der Autor auch an Aktivitäten im Kolleg und am Fachbereich beteiligt. Einen Überblick zeigt Abbildung 1.1. Beispielhaft sei hier die Mitorganisation des jährlichen Forschungskolleg-Symposiums „Future Trends in Service-Oriented Computing“, des „Server-Computing Summits“ am HPI, und die Beteiligung am Lehrbetrieb verschiedener Vorlesungen und Seminare genannt. Weiterhin schrieb der Autor zahlreiche Beiträge zum *Windows Research Kernel (WRK)* Blog „WRK@HPI“ [13], beispielsweise zur Implementierung neuer Systemaufrufe für den Windows Kern.

1.5 Struktur der Arbeit

In diesem Kapitel wurde der Kontext der Arbeit dargestellt und die Problemstellung erläutert. Weiterhin wurde der wissenschaftliche Beitrag der Arbeit zusammenfassend dargestellt.

Der verbleibende Teil der Arbeit ist wie folgt strukturiert:

Kapitel 2 stellt die Grundlagen der Arbeit dar. Konzepte auf dem Gebiet der Systeminstrumentierung und -überwachung werden vorgestellt und definiert.

Kapitel 3 beschreibt den im Rahmen dieser Arbeit entwickelten *Windows Monitoring Kernel* (WMK), einer Instrumentierungsinfrastruktur für Windows Systeme.

Kapitel 4 beschreibt die entwickelten Ansätze für die Online-Verarbeitung von Ereignissen durch Mustererkennung und Reaktionen.

Kapitel 5 zeigt anhand von Fallstudien, wie die entwickelten Konzepte praktisch eingesetzt werden können.

Kapitel 6 grenzt die Ergebnisse dieser Arbeit gegen verwandte Arbeiten ab und vergleicht die entwickelten Konzepte mit Alternativen.

Kapitel 7 fasst die Ergebnisse zusammen und bietet einen Ausblick auf weiterführende Fragestellungen.

Ergänzende Informationen zu den entwickelten Konzepten und Werkzeugen werden im Anhang gegeben.

1.6 Hinweise zur erstellten Software

Die im Rahmen dieser Arbeit entwickelten Konzepte und Algorithmen wurden innerhalb des *Windows Research Kernels (WRK)* implementiert und praktisch erprobt.

Im Sommer 2006 wurde der Quelltext des Betriebssystemkerns von Windows Server 2003 Enterprise Edition durch Microsoft für universitäre Einrichtungen zugänglich gemacht. Der Einsatz von Windows in Forschung und Lehre sollte dadurch verbessert werden [82].

Die Ergebnisse dieser Arbeit konnten in den Windows Kern integriert werden und dadurch mit praktisch relevanten Serversystemen evaluiert werden. Teile dieser Arbeit wurden durch Drittmittel von Microsoft unterstützt (Grant No. 15899).

Die entwickelten Konzepte und Algorithmen sind jedoch nicht Windows-spezifisch und lassen sich auf andere Betriebssysteme übertragen und in ähnlicher Form realisieren.

Dies gilt auch für das verwendete Instrumentierungssystem. In der Arbeit wurde der *Windows Monitoring Kernel (WMK)* entwickelt und verwendet. Die vorgestellten Konzepte zur Verarbeitung von Ereignissen sind ebenfalls auf andere Instrumentierungssysteme übertragbar.

1.7 Hinweise zu Sprache und Notation

Die Sprache der vorliegenden Arbeit ist Deutsch, daher wird versucht deutsche Begriffe zu verwenden. An Stellen, an denen dies nicht möglich ist, sei es, weil der deutsche Begriff missverständlich oder zu ungewöhnlich ist, oder sei es, weil es keine deutsche Entsprechung gibt, wird der englische Fachbegriff verwendet und *kursiv* dargestellt.

Bei Quellcode-Listings und Programmausgaben wurde auf eine Übersetzung verzichtet, das heißt Variablennamen und Kommentare werden auf Englisch dargestellt.