Gegenwärtige und zukünftige Entwicklung der Architekturen von Betriebssystemen

Michael Wittmann
Professionelle Textsatzsysteme
Technische Hochschule Ingolstadt
Matrikelnummer: 00098765
miw2006@thi.de

***Zusammenfassung*—Durch den zunehmenden Einsatz von Software in unterschiedlichen Umgebungen besteht Bedarf für angepasste Betriebssysteme, deren Eignung für den Anwendungszweck insbesondere von der Architektur der Betriebssysteme abhängt. Dieses Paper stellt die wichtigsten bestehenden Betriebssystemarchitekturen im Überblick vor und vergleicht deren Eigenschaften hinsichtlich Stabilität, Ausführungsgeschwindigkeit und Fehlerauswirkungen im Kernel. Anschließend werden aktuelle Trends in der Forschung verwendet, um die zukünftige Entwicklung der Betriebssystemarchitekturen zu prognostizieren. Die Untersuchung der gewählten Veröffentlichungen zeigt, dass monolithische Betriebssystemarchitekturen in mehreren Anwendungsbereichen zunehmend durch Mikrokernel-Architekturen und Kombinationen von Betriebssystemarchitekturen ersetzt werden. Die Betriebssystemarchitekturen werden dabei in Zukunft um Mechanismen erweitert werden, die die Sicherheit der Betriebssysteme erhöhen und die Betriebssysteme besser an die Anforderungen der Anwendungsbereiche anpassen.**

***Index Terms*—OS, operating system, software design, operating system structure**

I. Einleitung

Die Digitalisierung betrifft inzwischen zahlreiche Lebensbereiche und führt dazu, dass wir ständig von Computersystemen umgeben sind. Dies umfasst neben PCs für Büroarbeit auch Steuerrechner für Flug- und Fahrzeuge, Bankautomaten sowie Mobiltelefone und Computer in der Medizintechnik. Durch den Einsatz von Computern in unterschiedlichsten Umgebungen entstehen heterogene Anforderungen an die Systeme, die stark vom Nutzungszweck und den technischen und natürlichen Gegebenheiten des Umfeldes abhängen. Insbesondere die Auswahl des passenden Betriebssystems ist notwendig, um Anforderungen wie Echtzeitfähigkeit oder Sicherheitsstandards erfüllen zu können. Beispielsweise kann ein Betriebssystemabsturz eines PCs im Privatgebrauch akzeptiert werden. Im Fall eines Airbagsteuergeräts im Auto kann ein Absturz hingegen tödliche Folgen haben und ist somit inakzeptabel. Dieses Beispiel zeigt die Notwendigkeit eines für die Anforderungen geeigneten Betriebssystems. Bereits im Entwurf einer Betriebssystemarchitektur wird deshalb darauf geachtet, durch die Architektur Anforderungen an das Betriebssystem, z. B. Portabilität, zu erfüllen [1]. Aus diesem Grund ist die Betriebssystemarchitektur ein wichtiges Kriterium bei der Wahl des passenden Betriebssystems.

Im Folgenden soll die Frage beantwortet werden, wie sich die Betriebssystemarchitekturen in Zukunft entwickeln werden. Dazu werden zunächst die wichtigsten, bereits in der Praxis erprobten Architekturkonzepte vorgestellt und deren Merkmale und Vor- und Nachteile vergleichend gegenübergestellt. Anschließend werden aktuelle Forschungsergebnisse zur Architektur von Betriebssystemen herangezogen, um die zukünftige Evolution der Betriebssystemarchitekturen zu untersuchen.

II. Betriebssysteme und Softwarearchitektur

Um den Begriff der Betriebssystemarchitektur klar abzugrenzen, wird im Folgenden zunächst eine Arbeitsdefinition für die Betriebssystemarchitektur eingeführt. Anschließend wird erläutert, welchen Nutzen die Auswahl einer konkreten Betriebssystemarchitektur hat. Zuletzt wird das Konzept des Benutzermodus und Kernelmodus vorgestellt, weil es zum Verständnis der wesentlichen Unterschiede von Betriebssystemarchitekturen beiträgt.

A. Arbeitsdefinition der Betriebssystemarchitektur

In der Literatur wird der Begriff der Betriebssystemarchitektur meist nur grob definiert. Deshalb wird eine Arbeitsdefiniton mit den wichtigsten Aspekten mehrerer Autoren erarbeitet. So beschreibt Glatz die Architektur eines Betriebssystems als den strukturellen Aufbau der Software eines Betriebssystems [1]. Tanenbaum und Bos bestimmen den Begriff der Betriebssystemarchitektur näher als die innere Organisation des Betriebssystems [2]. Nach Mandl ist die Betriebssystemarchitektur eine Gliederung der Softwarebestandteile eines Betriebssystems in einzelne Komponenten und Module [3]. Damit stellt er die Einteilung des Betriebssystems in Funktionseinheiten in den Vordergrund. Weiterhin spezifiziert

BAUN den Begriff der Betriebssystemarchitektur genauer, indem er von der Architektur des Betriebssystemkerns (Kernel) spricht. Er unterscheidet Betriebssystemarchitekturen danach, welche Betriebssystemfunktionen im Kernel umgesetzt sind und welche nicht [4].

Aus den zuvor betrachteten Definitionsansätzen lässt sich die Arbeitsdefinition formulieren: Die Betriebssystemarchitektur beschreibt die innere Strukturierung der Betriebssystemsoftware in Einheiten und legt dadurch fest, welche Betriebssystemfunktionalitäten innerhalb oder außerhalb des Betriebssystemkernels realisiert werden.

B. Notwendigkeit verschiedener Betriebssystemarchitekturen

Wie bereits in Abschnitt I erwähnt, ergeben sich aus dem konkreten Einsatzzweck eines Betriebssystems unterschiedliche Anforderungen an den Entwurf des Betriebssystems. So kann die Entwicklung eines Betriebssystems beispielsweise darauf ausgerichtet sein, dass die Betriebssystemsoftware erweiterbar, skalierbar, portierbar oder echtzeitfähig ist [1]. Die Schwierigkeit im Architekturentwurf besteht darin, dass die Erfüllung einer Anforderung der Erreichung anderer Anforderungen entgegenstehen kann. Daher können nicht alle Anforderungen gleichzeitig erfüllt werden, sodass beim Entwurf der Architektur Kompromisse eingegangen werden müssen [1]. Diese Tatsache führt zu der Einsicht, dass die Betriebssystemarchitektur eine Grundlage für das Erfüllen der geforderten Eigenschaften des Betriebssystems ist.

C. Benutzermodus und Kernmodus

Das Betriebssystem eines Computers ist eine Softwaresammlung mit zwei grundlegenden Aufgaben: Es koordiniert den Zugriff auf Betriebsmittel wie Prozessor und Speicher zwischen mehreren Anwendungen und verdeckt den Hardwarezugriff, indem es den Anwendungen abstrahierte Schnittstellen zur Verfügung stellt [2]. Um diese Aufgaben uneingeschränkt übernehmen zu können, müssen unautorisierte Eingriffe der Anwendungen in das Betriebssystem verhindert werden [3]. Moderne Prozessoren bieten zu diesem Zweck Privilegierungsstufen an. Dabei können Prozesse innerhalb einer Privilegierungsstufe ihre Stufe nicht eigenmächtig verlassen und verfügen somit über begrenzte Rechte [4]. Obwohl einige Prozessoren über vier Privilegierungsstufen verfügen, nutzen Betriebssysteme typischerweise lediglich einen Benutzermodus und einen Kernmodus. Die Prozesse des Betriebssystemkernels werden im Kernmodus mit höheren Privilegien ausgeführt. Dadurch können Kernelprozesse direkt auf die Hardware und Kernelspeicherbereiche zugreifen [1]. Die übrigen Prozesse des Betriebssystems und alle Anwendungsprozesse laufen im Benutzermodus ab und dürfen somit nicht auf Code und Daten des Betriebssystemkernels zugreifen [3].

Beim Entwurf einer Betriebssystemarchitektur wird entschieden, welche Funktionalitäten des Betriebssystems in den Kernel integriert werden. Die restlichen Systemfunktionalitäten werden als Dienstprozesse aus dem Kernel ausgegliedert [4]. Die Betriebssystemfunktionen laufen dadurch entweder als privilegierte Prozesse im Kernmodus mit umfangreichen Berechtigungen oder als Dienstprozesse im Benutzermodus mit beschränkten Rechten ab. Diese Aufteilung ermöglicht die Isolation der Betriebssystemfunktionalitäten gegen gegenseitige Schadeinflüsse wie Datenmanipulation [1].

III. BESTEHENDE ARCHITEKTUREN IM ÜBERBLICK

Im Laufe der Zeit wurden viele Betriebssystemarchitekturen entwickelt. In diesem Abschnitt werden die grundlegenden Architekturkonzepte für Betriebssysteme vorgestellt. Danach werden die Architekturen hinsichtlich Stabilität, Ausführungsgeschwindigkeit und Fehlerauswirkungen im Kernel verglichen.

A. Monolithische Betriebssystemarchitekturen

Ein Betriebssystem besitzt eine monolithische Architektur, wenn alle Betriebssystemfunktionen Bestandteil des Kernels sind [4]. Abbildung 1 zeigt den grundlegenden Aufbau eines monolithischen Betriebssystems und die Zusammenarbeit zwischen dessen Komponenten. Um ein monolithisches Betriebssystem zu erzeugen, werden die Quellcode-Dateien der Teilkomponenten zuerst einzeln kompiliert und anschließend durch einen Linker zu einem einzigen großen Systemprogramm verbunden [2]. Dieses Vorgehen führt dazu, dass jede Funktion des Betriebssystemkernels sämtliche andere im Kernel verfügbaren Funktionen aufrufen kann [2]. Durch die fehlende Abgrenzung der Kernelkomponenten wird die Effizienz des Kernels erhöht [5]. Während die Möglichkeit zum Aufruf beliebiger Kernelfunktionen zunächst vorteilhaft erscheint, führt sie bei genauerer Betrachtung zu mehreren Problemen. So ist die Abfolge der Kernelfunktionsaufrufe nur schwer nachvollziehbar [2]. Schadsoftware mit Kernelzugriff, z. B. in Form eines Kernelmoduls für Linux, gefährdet die Sicherheit des Betriebssystems, weil sie fast unbeschränkt auf den gesamten monolithischen Kernel zugreifen kann [5]. Zudem kann ein Fehler in einer einzigen Kernelfunktion zum Absturz des gesamten Betriebssystemkernels führen [6].

Aus der Tatsache, dass der Kernel alle Funktionalitäten des Betriebssystems enthält, folgt nicht, dass monolithische Betriebssysteme keine innere Struktur besitzen können (siehe Abbildung 1). Häufig existiert beispielsweise ein zentraler Einstiegspunkt in den Kernel, der

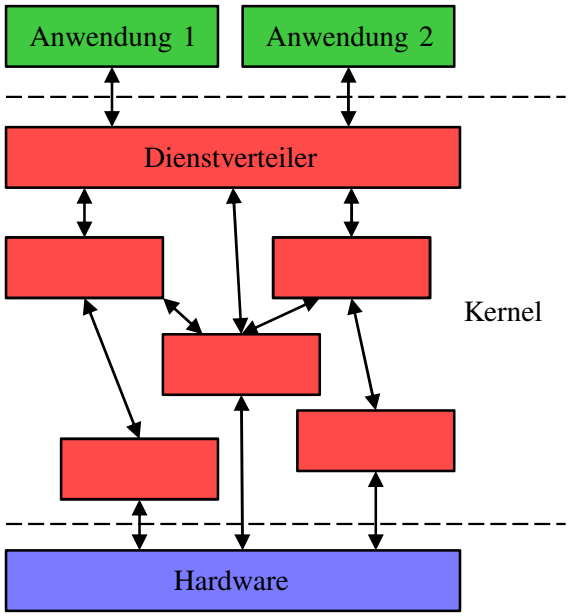

Abbildung 1. Monolithische Betriebssystemarchitektur [3]

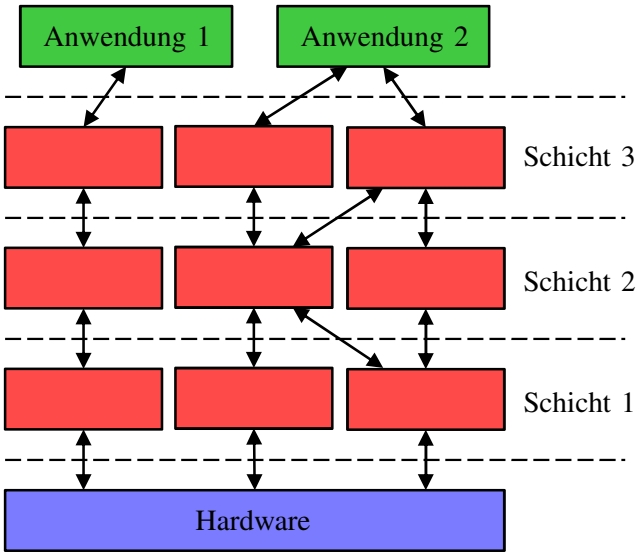

Abbildung 2. Geschichtete Betriebssystemarchitektur mit allen Schichten im Kernel [3]

Diensteverteiler [3]. Alle Systemaufrufe der Anwendungen treffen beim Diensteverteiler ein [3]. Der Diensteverteiler gibt die Systemaufrufe dann an die zuständigen Komponenten des Kernels zur Abarbeitung weiter [2][3]. Die Betriebssystemkomponenten, die im Fall einer monolithischen Architektur alle im Kernel vorhanden sind, bearbeiten die Systemaufrufe und kommunizieren mit der Hardware. In Abbildung 1 ist erkennbar, dass auch der Kernel einer monolithischen Architektur mit Diensteverteiler nur wenig innere Struktur besitzt.

Wie bereits in Unterabschnitt II-C beschrieben, führt der Prozessor alle im Kernel platzierten Betriebssystemfunktionen im Kernmodus, d. h. mit höheren Rechten, aus. Die erhöhten Privilegien des Kernmodus erlauben bei monolithischen Betriebssystemen somit allen Betriebssystemfunktionen, den gesamten Kerneladressraum zu nutzen [3]. Folglich kann ein fehlerhafter Kernelprozess unbeabsichtigt wichtige Daten im Kernelspeicher verändern. Außerdem kann ein von Angreifern übernommener, bösartiger Kernelprozess die Daten des Kernels absichtlich manipulieren oder stehlen. Da bei monolithischen Betriebssystemen alle Betriebssystemfunktionen im Kernel enthalten sind, kann jede der Funktionen den Kernel bei Vorliegen eines Fehlers zum Absturz bringen.

B. Geschichtete Architekturen

Bei geschichteten Architekturen werden die Funktionen des Betriebssystems auf mehrere Schichten aufgeteilt [3]. Eine Schicht umfasst alle Komponenten, die für eine bestimmte Funktionalität des Betriebssystems notwendig sind, z. B. alle Komponenten für den Hardwarezugriff [1]. Abbildung 2 stellt die Beziehungen zwischen benachbarten Schichten dar: Schicht n stellt der übergeordneten Schicht $n+1$ über eine Schnittstelle ihre Funktionalität zur Verfügung. Gleichzeitig nimmt Schicht n die Funktionalität von Schicht $n-1$ in Anspruch, indem Schicht n die Schnittstelle von Schicht $n-1$ nutzt.

Es gibt keine allgemeingültigen Vorschriften, wie viele Schichten existieren und in welche Schicht eine Betriebssystemfunktion eingeordnet wird [1]. Daher kann die Zuordnung der Betriebssystemfunktionen zu den Schichten in verschiedenen Betriebssystemen unterschiedlich gestaltet sein. Wie aus Abbildung 2 ersichtlich, sind die Schichten hierarchisch angeordnet [2]. Die Betriebssystemfunktionen in Schicht n greifen ausschließlich auf Betriebssystemfunktionen in der direkt darunterliegenden Schicht $n-1$ zu [3]. Dementsprechend durchlaufen Systemaufrufe der Anwendungen alle Schichten des Betriebssystems hinunter zur Hardware und Eingabedaten den umgekehrten Weg durch alle Schichten bis zur Anwendung [1]. In manchen Fällen greift eine Schicht dennoch auf Schichten unterhalb der Vorgängerschicht zu, um z. B. eine deutliche Geschwindigkeitssteigerung zu erreichen, weil die dazwischenliegenden Schichten nicht durchlaufen werden müssen [3].

Üblicherweise sind bei geschichteten Betriebssystemen alle Schichten im Kernel enthalten (vgl. rot gefärbte Komponenten in Abbildung 2) [2]. In diesem Fall sind alle Betriebssystemfunktionen, wie bei monolithischen Betriebssystemen, im Kernel umgesetzt. Durch die Aufteilung in Schichten könnten die oberen Schichten des Betriebssystems aber auch auf die Benutzerebene verschoben werden, um den Umfang des Kernels zu verkleinern.

C. Mikrokernel-Architekturen

Bei Mikrokernel-Architekturen wird das Betriebssystem in genau abgegrenzte Komponenten aufgeteilt [2].

Die wichtigste Komponente, der Betriebssystemkernel, übernimmt nur die essentiellen Grundaufgaben des Betriebssystems und wird deswegen Mikrokernel genannt. Der Mikrokernel ist nur für die Verwaltung des Speichers und der Prozesse, die Synchronisation und die Steuerung der Interprozesskommunikation zuständig [4]. Die im Mikrokernel enthaltenen Funktionen werden wegen ihrer Kernelzugehörigkeit als Prozesse im Kernmodus (siehe Unterabschnitt II-C) ausgeführt [2]. Alle weiteren Komponenten des Betriebssystems, die nicht in den Mikrokernel integriert sind, laufen als isolierte Serverprozesse im Benutzermodus mit geringeren Rechten ab [2][7]. Die Serverprozesse des Betriebssystems sind somit Benutzerprozesse, bei denen ein Absturz lediglich die Funktionstüchtigkeit der zugehörigen Betriebssystemkomponente beeinträchtigt, aber nicht zum Absturz des gesamten Betriebssystems führt [2]. Beispiele für Komponenten außerhalb des Mikrokernels sind Gerätetreiber, Dateisystemserver und Netzwerkserver [1][3].

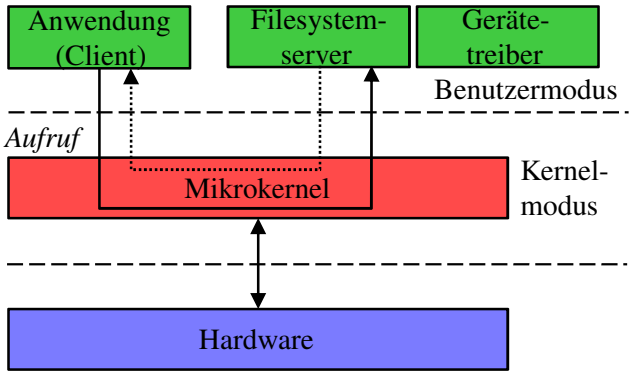

Abbildung 3. Systemaufruf in Mikrokernel-Betriebssystemarchitektur [1][3]

Die Diensterbringung in Mikrokernel-Betriebssystemen funktioniert nach dem Client-Server-Prinzip [1]. Um die Dienste des Betriebssystems in Anspruch zu nehmen, führt eine Nutzeranwendung (Client) einen Systemaufruf beim Mikrokernel durch (siehe Abbildung 3). Der Mikrokernel gibt den Systemaufruf an den Serverprozess (Server) weiter, der für die Diensterbringung verantwortlich ist. Alle Serverprozesse werden dabei wie die Nutzeranwendung im Benutzermodus ausgeführt. Der aktivierte Serverprozess liefert das Ergebnis des Systemaufrufs an den Mikrokernel zurück, welcher das Ergebnis schließlich der Nutzeranwendung mitteilt [1][3].

Die Mikrokernel-Architektur stellt einen Gegenentwurf zur monolithischen Architektur dar, weil ein Mikrokernel auf die minimal notwendigen Betriebssystemfunktionen beschränkt ist. Dagegen enthält der Kernel eines monolithischen Betriebssystems einen Großteil aller Betriebssystemfunktionen, die als Kernelfunktionen im privilegierten Kernmodus ausgeführt werden [8].

D. Hybride Architekturen

Die Architekturen besitzen aufgrund ihrer Eigenschaften verschiedene Vor- und Nachteile. Betriebssysteme mit Mikrokernel sind beispielsweise langsamer als Betriebssysteme mit anderen Architekturen, weil öfter zwischen den Prozessen umgeschaltet wird [4]. Betriebssysteme mit hybriden Architekturen versuchen daher, die Vorteile von monolithischen Architekturen mit denen von Mikrokernel-Architekturen zu verbinden [4]. Damit stellt die hybride Betriebssystemarchitektur eine Mischung von monolithischer und Mikrokernel-Architektur dar. Ein hybrider Kernel enthält mehr Funktionalitäten als ein klassischer Mikrokernel [4]. Dadurch soll erreicht werden, dass ein Betriebssystem mit hybridem Kernel eine höhere Ausführungsgeschwindigkeit als ein mikrokernelbasiertes Betriebssystem erreicht. Zudem ist ein Betriebssystem mit hybridem Kernel stabiler als ein monolithisches Betriebssystem, weil es das grundlegende Konzept des Mikrokernels beibehält [4].

E. Vergleich ausgewählter Architekturen

Aus den zuvor vorgestellten Architekturen werden die monolithische Architektur, die Mikrokernel-Architektur und die hybride Architektur für die Gegenüberstellung herangezogen. Geschichtete Architekturen sind im Wesentlichen eine verbesserte Variante der monolithischen Architekturen [3] und werden deswegen für den Vergleich nicht betrachtet. Im Folgenden wird die Zuordnung der Betriebssystemfunktionen zu Kern- und Benutzermodus für die ausgewählten Architekturen in einer Übersicht dargestellt. Anschließend werden die monolithische Architektur und die Mikrokernel-Architektur hinsichtlich Stabilität, Ausführungsgeschwindigkeit und Fehlerauswirkungen im Kernel verglichen.

Abbildung 4 zeigt für monolithische, hybride und mikrokernelbasierte Architekturen, welche Betriebssystemkomponenten außerhalb des Kernels im Benutzermodus oder als Kernelbestandteil im Kernmodus ausgeführt werden. Bei monolithischen Betriebssystemen sind sämtliche Betriebssystemfunktionen in den Kernel integriert, d. h. das komplette Betriebssystem läuft im Kernmodus. Im Gegensatz dazu enthält ein Betriebssystem mit Mikrokernel-Architektur nur die notwendigsten Funktionen im Kernel. Alle weiteren Komponenten des Betriebssystems werden vom Kernel in Benutzerprozesse verschoben (siehe Tabelle I). Betriebssysteme mit hybrider Architektur stellen eine Zwischenstufe dar. Bei diesem Konzept beinhaltet der Kernel neben den zentralen Funktionen des Mikrokernels zusätzlich weitere Betriebssystemkomponenten. Durch deren Integration in den Kernel ist eine Performancesteigerung gegenüber Betriebssystemen mit Mikrokernel möglich [4].

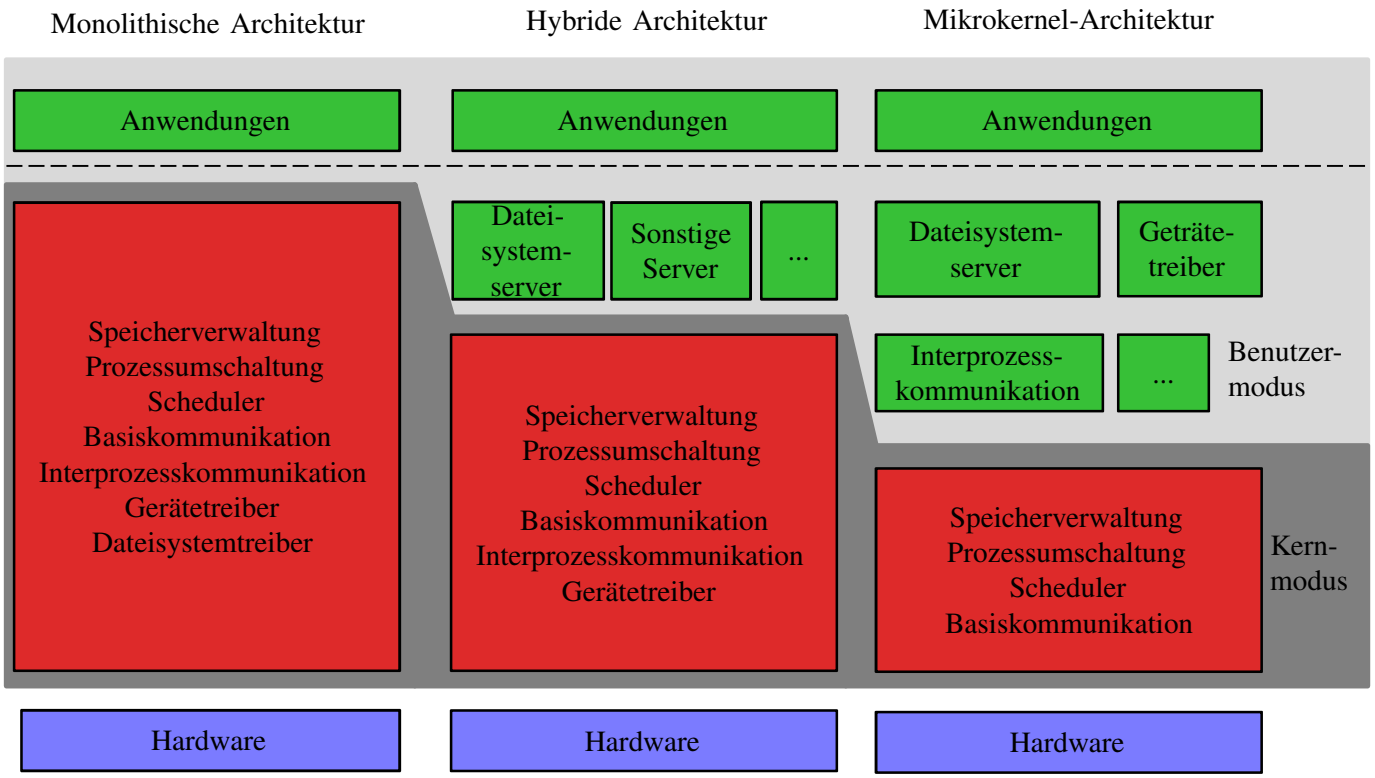

Abbildung 4. Vergleich ausgewählter Betriebssystemarchitekturen [3][4]

Tabelle I. Privilegienstufen der Prozesse in verschiedenen Architekturen

	Betriebssystemarchitektur	
Prozesse	**Monolithisch**	**Mikrokernel**
Anwendungen	Benutzermodus	Benutzermodus
zentrale Systemfunktionen	Kernmodus	Kernmodus
sonstige Systemfunktionen	Kernmodus	Benutzermodus

Im Anschluss an die Gegenüberstellung der ausgewählten Architekturen wird nun aufgezeigt, inwiefern die verschiedenen Architekturen Stabilität, Ausführungsgeschwindigkeit und Fehlerauswirkungen im Kernel beeinflussen [1]. Dabei werden ausschließlich die monolithische und die Mikrokernel-Architektur verglichen, weil diese Grundarchitekturen die Extrema der Architekturformen darstellen:

- Auf der einen Seite die monolithischen Betriebssysteme, die alle Betriebssystemkomponenten im Kernel enthalten und
- auf der anderen Seite die Mikrokernel-Betriebssysteme, deren Mikrokernel auf die minimal notwendigen Komponenten reduziert sind.

Andere Architekturen, z. B. die geschichtete und die hybride Architektur, sind Abwandlungen oder Zwischenstufen der beiden Grundarchitekturen und werden deswegen nicht weiter betrachtet.

1) Stabilität: Mikrokernel-Betriebssysteme sind aufgrund der geringeren Zahl an Prozessen im Kernmodus prinzipiell stabiler als monolithische Betriebssysteme. Da viele monolithische Betriebssysteme oft über einen längeren Zeitraum gewachsen sind, bieten sie dennoch oft eine ähnliche Stabilität wie Mikrokernel-Betriebssysteme [4].

2) Ausführungsgeschwindigkeit: Aufgrund der Auslagerung von Betriebssystemfunktionen in Benutzerprozesse sind bei Mikrokernel-Betriebssystemen mehr Prozesswechsel notwendig als bei monolithischen Betriebssystemen [4][9]. Daraus folgt, dass Mikrokernel-Betriebssysteme wegen ihrer Architektur langsamer als Betriebssysteme mit anderen Architekturen sind. Mit dem Mikrokernel L4 konnte jedoch gezeigt werden, dass auch ein Mikrokernel durch Optimierungen nur geringe Geschwindigkeitseinbußen verursachen kann [8].

3) Fehlerauswirkungen im Kernel: Bei monolithischen Betriebssystemen beinhaltet der Kernel alle Komponenten des Betriebssystems, d. h. alle Komponenten werden im Kernmodus mit höheren Rechten ausgeführt [4]. Dadurch kann eine fehlerhafte Komponente den gesamten Kernel und somit das Betriebssystem zum Absturz bringen [6]. Bei Mikrokernel-Betriebssystemen hingegen werden möglichst viele Komponenten aus dem Kernel in den Benutzermodus verschoben (siehe Tabelle I). Falls eine dieser ausgelagerten Komponenten fehlerhaft ist, hat sie wegen der Einschränkungen des

Benutzermodus keinen Zugriff auf den Mikrokernel und kann diesen folglich nicht zum Absturz bringen [6]. Stattdessen kann ein separater Dienstprozess (reincarnation server) fehlerhafte Komponenten im Benutzermodus erkennen und gegebenenfalls ersetzen [2].

IV. Aktuelle Forschungsthemen

Als Grundlage für die Prognose der zukünftigen Entwicklung der Betriebssystemarchitekturen in Abschnitt V werden aktuelle Forschungspapiere herangezogen. Dazu werden ausschließlich Beiträge mit Bezug zu Betriebssystemarchitekturen betrachtet, die seit 2015 veröffentlicht wurden. Im Folgenden werden die Inhalte der Forschungspapiere nach Themen gruppiert vorgestellt.

A. Verbesserung monolithischer Architekturen

In [10] wird eine Betriebssystemarchitektur, die auf dem neuen Prinzip des Nested Kernel beruht, vorgestellt. Dazu wird der monolithische Kernel in zwei Bereiche, Nested Kernel und Outer Kernel, aufgeteilt. Der Nested Kernel ist ein kleiner, isolierter Teil des monolithischen Kernels und der Kerneldaten, der vollen Systemzugriff besitzt und die Speicherverwaltungseinheit MMU kontrolliert. Der übrige Kernel wird als Outer Kernel bezeichnet und in der Architektur als unvertrauenswürdig angesehen. Laut den Autoren kann durch diese Aufteilung die Sicherheit eines monolithischen Kernels unter geringen Geschwindigkeitsverlusten erhöht werden [10].

B. Betriebssystemarchitekturen für Mehrkernsysteme

In [11] wird die Verteilung der Ausführung des Betriebssystems und der Anwendungen auf Prozessoren mit mehreren Kernen untersucht. Dabei wird die Verteilung unter Nutzung des monolithischen Betriebssystems Linux und des für Mehrkernsysteme entwickelte Mikrokernel-Betriebssystems AnT verglichen. In Mikrokernel-Betriebssystemen enthält der Mikrokernel nur die notwendigsten Komponenten, die übrigen Betriebssystemdienste werden als Prozesse auf Benutzerebene realisiert (siehe Tabelle I). Da die Dienstprozesse als Benutzerprozesse außerhalb des Kernels umgesetzt sind, können die Berechnungen für Betriebssystemdienstprozesse und Anwendungsprozesse auf verschiedene Kerne verteilt werden. Bei monolithischen Betriebssystemen hingegen muss das Betriebssystem auf demselben Kern ausgeführt werden wie die aufrufende Anwendung, da die Betriebssystemfunktion durch den Systemaufruf der Anwendung gestartet wird. Dadurch kann die Ausführung der Betriebssystemfunktionen nur auf mehrere Kerne verteilt werden, wenn die Anwendungsprozesse auf mehrere Kerne verteilt werden. Ein Vergleich zwischen AnT und Linux zeigt, dass Mikrokernel-Betriebssysteme monolithische Betriebssysteme bei der Performance übertreffen können, wenn die Berechnungen auf mehreren Kernen verteilt sind [11].

Auch [6] behandelt die Unterstützung von Mehrkernprozessoren durch die Betriebssystemarchitektur. Die Autoren argumentieren, dass ein fehlertolerantes, hochzuverlässiges Betriebssystem die Möglichkeiten von Mehrkernprozessoren am besten nutzt. Um die passende Betriebssystemarchitektur für Mehrkernprozessoren auszuwählen, werden die Eigenschaften der monolithischen und mikrokernelbasierten Architektur untersucht. Nach [6] ermöglichen Mikrokernel-Betriebssysteme aufgrund ihres modularen Aufbaus höhere Sicherheit, Zuverlässigkeit und Flexibilität und sind somit besser für Mehrkernprozessoren geeignet als monolithische Betriebssysteme.

C. Betriebssystemarchitekturen für Big Data und Cloud Computing

XOS ist ein Betriebssystem für Rechenzentren, das die Anwendungen ihre Ressourcen selbst verwalten lässt [12]. Der Kernel stellt lediglich den Schutz und die Aufteilung der Ressourcen für einen mehrfachen Zugriff sicher. Die Architektur von XOS greift den Ansatz der Mikrokernel-Architektur, den Kernel zu minimieren, auf. Darüber hinaus wird auch ein Teil der Ressourcenverwaltung aus dem Kernel entfernt. XOS lässt dadurch die Anwendungen ihre Ressourcen selbst konfigurieren und einteilen. Da der Kernel von XOS aus Kernelmodulen für Linux besteht, ist die Betriebssystemarchitektur eine Weiterentwicklung der monolithischen Architektur von Linux [12].

Das für Big Data entwickelte Betriebssystem NileOS basiert auf einer verteilten Mikrokernelarchitektur [13]. Das Gesamtsystem ist ein verteiltes System mit mehreren per Netzwerk verbundenen Knoten. Auf jedem Prozessorkern eines Knotens läuft ein eigener Mikrokernel. Dabei wird jedem Kern eine Rolle zugeordnet, die dessen verfügbare Dienste bestimmt. Dadurch können Berechnungen asymmetrisch auf verschiedene Kerne verteilt werden [13].

Um verschiedenartige Hardware in Rechenzentren zu nutzen, wird in [14] ein Rack-Betriebssystem erarbeitet. Dazu werden verschiedene Mikrokernel-Betriebssysteme, die unterschiedliche Zielsetzungen haben, kombiniert. Das Rack-Betriebssystem besteht aus verschiedenen Mikrokernen für die verschiedenen Recheneinheiten und einem Rack-Kernel für die Kommunikation zwischen den Boards der Recheneinheiten. Es werden keine monolithische Betriebssysteme verwendet, weil deren Sicherheit aufgrund des umfangreichen Kernels schwer sicherzustellen ist [14].

D. Sicherheit durch Architektur

In [9] wird betrachet, wie die Sicherheit eines **B**uilding **A**utomation **S**ystems (BAS) durch die Betriebssystemar-

chitektur erhöht werden kann. Ein BAS ist ein cyber-physisches System zur Steuerung von Gebäudefunktionen wie Beleuchtung, Heizung und Belüftung. Dabei werden sowohl kritische als auch unkritische, mit dem Netzwerk verbundene Anwendungen auf einem Betriebssystem ausgeführt. Deshalb wird eine Möglichkeit gesucht, die kritischen Anwendungen durch die Betriebssystemarchitektur gegen bösartige Einwirkungen von durch Dritte übernommene, nicht kritische Anwendungen zu schützen. Um die passende Betriebssystemarchitektur zu finden, werden eine erweiterte Variante des Mikrokernelbetriebssystems MINIX 3, der formal verifizierte Mikrokernel seL4 und das monolithische Betriebssystem Linux verglichen. In zwei Experimenten wird simuliert, dass ein Angreifer einen unkritischen Anwendungsprozess übernimmt und beliebigen Code ausführen kann und dass dieser Prozess Root-Berechtigungen besitzt. Bei den Mikrokerneln MINIX 3 und seL4 zeigen sich keine Auswirkungen der Angriffe auf kritische Anwendungen wie eine Temperatursteuerung. Bei Linux hingegen können kompromittierte, unkritische Anwendungen die kritischen Anwendungen beeinträchtigen und so die Sicherheit bedrohen. Daraus folgern die Autoren, dass eine Betriebssystemarchitektur mit Mikrokernel als Grundlage für sicherere und fehlertolerante cyber-physische Systeme verwendet werden kann [9].

In [7] wird das **T**rusted **E**xecution **E**nvironment (TEE) Betriebssystem MicroTEE auf Basis einer Mikrokernelarchitektur vorgestellt. Bestehende TEE-Betriebssysteme besitzen meist eine monolithische Architektur. Aufgrund dessen könnte das gesamte TEE-Betriebssystem kompromittiert werden, wenn ein beliebiger Kerneldienst Schwachstellen aufweist. Der Mikrokernel von MicroTEE hingegen führt nur die zentralsten Aufgaben des Betriebssystems aus, andere wichtige Systemdienste werden als Prozesse auf Benutzerebene voneinander isoliert. Dadurch kann eine Schwachstelle in einer Komponente nicht zur Kompromittierung des gesamten Betriebssystems führen, solange der Mikrokernel nicht kompromittiert ist [7].

Die monolithische und die mikrokernelbasierte Architektur werden in [8] hinsichtlich ihrer Anfälligkeit für Schwachstellen verglichen. Dazu wird für bekannte, kritische Schwachstellen des monolithischen Betriebssystems Linux untersucht, ob diese Schwachstellen durch eine Mikrokernel-Architektur verhindert oder deren Auswirkungen durch einen Mikrokernel abgeschwächt werden können. Die Untersuchung zeigt, dass nur 4 % der betrachteten Schwachstellen bei einer Mikrokernel-Architektur ebenfalls kritische Auswirkungen hätten und 40 % der Schwachstellen durch ein Betriebssystemdesign mit formal verifiziertem Mikrokernel vollständig verhindert werden könnten [8].

V. Zukünftige Entwicklung der Betriebssystemarchitekturen

Die in Abschnitt IV vorgestellten Forschungsbeiträge behandeln Betriebssystemarchitekturen in verschiedenen Anwendungsbereichen, z. B. Cloud Computing oder Personal Computer. Aus diesen Anwendungsbereichen ergeben sich unterschiedliche Anforderungen an die Betriebssysteme und deren Architektur. Dementsprechend wird die Prognose der zukünftigen Entwicklung der Betriebssystemarchitekturen in Unterabschnitte, die unterschiedliche Anforderungen und Entwicklungsrichtungen abbilden, unterteilt. Im Folgenden wird zuerst die mögliche Entwicklung der Betriebssystemarchitekturen für Personal Computer, Mehrkernsysteme, Cloud Computing und Big Data sowie für Systeme mit hohen Sicherheitsanforderungen prognostiziert. Anschließend wird zusammenfassend dargestellt, in welche Richtung sich Betriebssystemarchitekturen im Allgemeinen entwickeln werden.

A. Entwicklung der Architektur bei Personal Computer

Unter Personal Computer (PC) sind in diesem Zusammenhang Rechner wie Desktopcomputer, Notebooks, Tablets und Smartphones zu verstehen. Auf diesen Geräten laufen aktuell monolithische Betriebssysteme wie Windows, Linux und Mac OS, die für die Interaktion mit Benutzern [13], hohe Geschwindigkeit und Funktionalität entworfen wurden [8]. Da monolithische Betriebssysteme die Anforderungen an Benutzerinteraktion, Geschwindigkeit und Funktionalität weiterhin erfüllen, ist aus Sicht dieser Anforderungen keine andere Betriebssystemarchitektur notwendig.

Durch die zunehmende Speicherung schützenswerter Daten auf PCs wird die Sicherheit für PCs immer wichtiger, aber monolithische Betriebssysteme verfügen nicht über ausreichende Sicherheitsmechanismen [10]. Zudem begünstigt die monolithische Architektur aufgrund des umfangreichen und komplexen Kernels Schwachstellen, die die Sicherheit monolithischer Betriebssysteme beeinträchtigen [8]. Aktuelle Forschungsbeiträge, z. B. der Nested Kernel (siehe Unterabschnitt IV-A) zeigen, dass die Sicherheit monolithischer Betriebssysteme verbessert werden kann, ohne die monolithische Architektur aufzugeben.

Aufgrund ihrer Eignung für die zuvor beschriebenen Anforderungen werden in Zukunft weiter monolithische Betriebssystemarchitekturen für PCs eingesetzt werden. Um die architekturbedingten Sicherheitsmängel zu begrenzen, werden monolithische Betriebssystemarchitekturen um architekturbasierte Sicherheitsmaßnahmen wie den Nested Kernel [10] erweitert werden.

B. Entwicklung der Architektur bei Mehrkernsystemen

Systeme mit Mehrkernprozessoren verfügen über mehrere Recheneinheiten auf einem Chip. Insbesondere die effiziente Verteilung der Berechnungen auf die Prozessorkerne muss durch das Betriebssystem gewährleistet werden.

Die in Unterabschnitt IV-B beschriebene Gegenüberstellung des Mikrokernel-Betriebssystems AnT und des monolithischen Betriebssystems Linux zeigt, dass sich Mikrokernelarchitekturen besser für die Anforderungen von Mehrkernprozessoren eignen, weil Betriebssystem- und Anwendungsprozesse wegen des Mikrokernels besser über die Kerne verteilt werden können [11]. Weiterhin sind Mikrokernel-Betriebssysteme aufgrund ihrer Architektur sicherer, flexibler und zuverlässiger als monolithische Betriebssysteme [6].

Die in Unterabschnitt IV-B vorgestellten Forschungsergebnisse deuten darauf hin, dass sich Mikrokernel-Betriebssysteme wegen ihrer architekturbedingten Eigenschaften besser für Mehrkernsysteme eignen als monolithische Betriebssysteme. Im Bereich der Mehrkernsysteme ist deshalb ein zunehmender Einsatz von Mikrokernel-Betriebssystemen wahrscheinlich. Allerdings verweist [15] darauf, dass einige Fragen zum Einsatz von Mikrokernel-Architekturen in Mehrkernsystemen noch nicht final geklärt sind. So ist z. B. fraglich, welche Kerneldatenstrukturen zwischen Prozessorkernen synchronisiert werden müssen. Außerdem ist festzulegen, welche Dienste innerhalb des Kernels, als ausgelagerte Dienstprozesse oder direkt in den Anwendungen umgesetzt werden und welche Auswirkungen diese Zuordnung hat [15]. Insgesamt werden Mikrokernel-Architekturen für Mehrkernsysteme angepasst werden, indem die Organisation der Kerneldaten und die Platzierung von Dienstprozessen innerhalb oder außerhalb des Kernels für Mehrkernsysteme optimiert werden.

C. Entwicklung der Architektur bei Big Data und Cloud Computing (Mehrprozessorsysteme)

Für Big Data und Cloud Computing werden Systeme mit mehreren Prozessoren (Mehrprozessorsysteme) verwendet. Damit grenzen sich diese Systeme von Mehrkernsystemen, die einen Prozessor mit mehreren Kernen besitzen, ab. Betriebssysteme für Big Data und Cloud Computing müssen die Verteiltheit des Systems über mehrere Prozessoren unterstützen und die Kommunikation zwischen den verteilten Einheiten ermöglichen. Zudem sollten die Betriebssysteme flexibel sein, um die Vorteile heterogener Hardwarebausteine nutzen zu können. Darüber hinaus ist die Sicherheit des Betriebssystems, z. B. in Cloud-Rechenzentren mit Internetverbindung, wichtig.

Für das Rack-Betriebssystem (siehe Unterabschnitt IV-C) wurde eine Mikrokernel-Architektur gewählt, weil Mikrokernel eine höhere Sicherheit bieten und Mikrokernel-Architekturen Fehlerauswirkungen besser eindämmen als monolithische Architekturen [14]. Bei NileOS wird keine monolithische Architektur eingesetzt, weil diese die Erweiterbarkeit des Betriebssystems erschwert [13]. Stattdessen wird eine verteilte Betriebssystemarchitektur mit verteilten Mikrokerneln auf den einzelnen Knoten eingesetzt. Diese Mikrokernel können leicht um zusätzliche Dienste erweitert werden [13]. Die Architektur von XOS ist aus einer monolithischen Architektur hervorgegangen, indem der Kernel wie in Mikrokernel-Architekturen verkleinert wurde [12].

Es ist somit erkennbar, dass sich die Betriebssystemarchitekturen für Big Data und Cloud Computing in Zukunft zur verteilten Architektur und zur Mikrokernel-Architektur hin entwickeln werden. Die Kombination von verteilter und mikrokernelbasierter Architektur bietet die Möglichkeit, verteilte Systeme durch ein gemeinsames Betriebssystem zu verwalten und dabei die Vorteile der Mikrokernel auf den einzelnen Knoten zu nutzen. Der Trend, möglichst viele Betriebssystemfunktionen aus dem Kernel auszulagern, um kleinere Kernel mit höherer Sicherheit und Flexibilität zu erhalten, wird die Entwicklung in Richtung von Mikrokernel-Architekturen lenken.

D. Entwicklung der Architektur bei Systemen mit hohen Sicherheitsanforderungen

Einige Anwendungsbereiche stellen hohe Anforderungen an die Sicherheit des eingesetzten Betriebssystems. Dabei stehen der Schutz des Betriebssystemkernels gegen Bedrohungen und die Isolation von Prozessen durch die Betriebssystemarchitektur im Vordergrund.

Beim Entwurf eines Building Automation Systems (siehe Unterabschnitt IV-D) zeigt sich, dass Betriebssysteme mit mikrokernelbasierten Architekturen kritische Prozesse besser gegen Angriffe schützen als ein Betriebssystem mit monolithischer Architektur [9]: Im monolithischen Betriebssystem Linux kann ein bösartiger Prozess sicherheitskritische Prozesse durch Interprozessnachrichten manipulieren und mit Root-Rechten kritische Prozesse beenden. Durch eine Kontrollkomponente im Kernel kann ein Mikrokernel hingegen die Interprozesskommunikation überwachen und schädliche Nachrichten unterbinden [9]. In [7] wird das Trusted Execution Environment Betriebssystem MicroTEE auf einem Mikrokernel aufgebaut. Die Mikrokernel-Architektur stellt die Isolation der Anwendungen auf Benutzerebene sicher. Dadurch bleibt die Sicherheit des gesamten Betriebssystems bei einem Problem in einer Komponente gewahrt, solange der Mikrokernel nicht kompromittiert ist. Da die Mikrokernel-Architektur einen formalen Beweis der Fehlerfreiheit des Mikrokernels ermöglicht, wird

zudem die Sicherheit des MicroTEE-Betriebssystems durch den eingesetzten seL4-Mikrokernel erhöht. Die Untersuchung der Auswirkungen von Schwachstellen in monolithischen und mikrokernelbasierten Betriebssystemen zeigt ebenfalls, dass eine Mikrokernel-Architektur die Sicherheit des Betriebssystems erhöht [8]. Allein durch die Verwendung eines Mikrokernels anstelle eines monolithischen Kernels würden 29 % der für ein monolithisches Betriebssystem kritischen Schwachstellen vollständig verhindert. Mit einem formal verifizierten Mikrokernel würden weitere 11 % der Schwachstellen komplett vermieden. Insgesamt würden 40 % der untersuchten kritischen Schwachstellen vollständig unterbunden. 17 % der für ein monolithisches Betriebssystem kritischen Schwachstellen würden bei einer Mikrokernel-Architektur lediglich geringe und weitere 38 % keine kritischen Auswirkungen haben, sodass nur 4 % der betrachteten Schwachstellen auch bei einer Mikrokernel-Architektur als kritisch eingestuft würden [8].

Wie [7], [8] und [9] belegen, ermöglichen Mikrokernel-Architekturen durch den minimalen Kernel eine höhere Sicherheit als monolithische Architekturen. Für Systeme mit hohen Sicherheitsanforderungen werden deshalb Mikrokernel-Architekturen eingesetzt werden. Die Mikrokernel-Architekturen werden dabei in Zukunft um Mechanismen für eine stärkere Isolation zwischen den Anwendungsprozessen erweitert werden, wie die Komponente zur Kontrolle der Interprozesskommunikation [9] beispielhaft zeigt. Darüber hinaus werden zukünftig vermehrt Mikrokernel-Betriebssysteme mit formal bewiesenem Mikrokernel entwickelt werden, um Sicherheitseigenschaften der Mikrokernel-Architektur nachweisen zu können.

E. Zusammenfassung der Entwicklungstendenzen

Bei der Analyse der Forschungsbeiträge in den vorhergehenden Unterabschnitten zeichnet sich ab, dass zunehmend unterschiedliche Betriebssystemarchitekturen eingesetzt werden. Dazu gehören neben den monolithischen und mikrokernelbasierten auch verteilte Architekturen und Architekturkombinationen, die für einen Einsatzzweck Elemente verschiedener Betriebssystemarchitekturen verbinden.

Auf PCs werden weiterhin Betriebssysteme mit monolithischen Architekturen verwendet werden. Dabei wird die vergleichsweise geringe Sicherheit, die durch die monolithische Architektur bedingt ist, durch architekturbasierte Sicherheitsmaßnahmen wie dem Nested Kernel [10] erhöht werden. Bei Mehrkernsystemen hingegen werden in Zukunft Mikrokernel-Betriebssysteme eingesetzt werden, deren Kerneldatenverwaltung und Systemdienstverteilung für die Nutzung mehrerer Prozessorkerne optimiert sind. Auch für Cloud Computing und Big Data werden weniger monolithische Betriebssysteme zum Einsatz kommen. Stattdessen werden die Betriebssystemarchitekturen eine Kombination der verteilten und mikrokernelbasierten Architektur darstellen, die auf die einzelnen Knoten des Gesamtsystems Mikrokernel verteilt. Durch die Ausrichtung auf Benutzerinteraktion [13] und die Platzierung der meisten Betriebssystemkomponenten im Kernel eignen sich monolithische Betriebssysteme wegen ihrer Architektur nicht für Einsatzbereiche mit speziellen Sicherheitsanforderungen. Insbesondere in Systemen mit hohen Sicherheitsanforderungen wird deshalb die Nutzung von Mikrokernel-Architekturen zunehmen. Ein Grund hierfür ist die Robustheit von Mikrokernelbetriebssystemen, die durch eine verringerte Anzahl von Funktionen im Kernmodus erreicht wird [9]. Zur besseren Absicherung werden für diesen Einsatzbereich Mikrokernel entwickelt werden, deren Sicherheitseigenschaften formal bewiesen werden. Zudem werden die Mikrokernel um Maßnahmen zur strikten Isolation von Benutzerprozessen erweitert werden.

VI. Conclusion

Die Architektur eines Betriebssystems bestimmt wichtige Eigenschaften des Betriebssystems und beeinflusst daher, ob das Betriebssystem den Anforderungen einer Zielumgebung an ein Betriebssystem genügt. Deshalb wurde angesichts des zunehmenden Einsatzes von Betriebssystemen in Umgebungen mit unterschiedlichen Anforderungen untersucht, wie sich die Betriebssystemarchitekturen in Zukunft entwickeln werden. Dazu wurden zunächst die grundlegenden Architekturkonzepte vorgestellt und verglichen. Die anschließende Analyse gegenwärtiger Entwicklungen im Bereich der Betriebssystemarchitekturen ergab, dass Verbesserungen einer Betriebssystemarchitektur meist der Anpassung an spezielle Anforderungen eines konkreten Einsatzbereichs dienen. Für verschiedene Anwendungsbereiche wurde erarbeitet, welche Architekturen dort zukünftig eingesetzt werden und wie diese Architekturen zukünftig erweitert werden, damit sie sich für ihren Anwendungsbereich besser eignen. Insgesamt zeigt der Vergleich der Veröffentlichungen, dass in Zukunft neben monolithischen vermehrt auch Mikrokernel-Architekturen, verteilte Architekturen und Architekturkombinationen verwendet werden.

Literatur

[1] Eduard Glatz. *Betriebssysteme: Grundlagen, Konzepte, Systemprogrammierung*. 4. Auflage. Heidelberg: Dpunkt.verlag, 2019. ISBN: 978-3-96088-839-0.

[2] Andrew S. Tanenbaum und Herbert Bos. *Modern operating systems*. 4. Auflage. Boston (Massachusetts): Pearson Education, 2015. ISBN: 978-0-13-359162-0.

[3] Peter Mandl.
Grundkurs Betriebssysteme: Architekturen, Betriebsmittelverwaltung, Synchronisation, Prozesskommunikation, Virtualisierung.
5. Auflage. Wiesbaden: Springer Fachmedien Wiesbaden und Imprint: Springer Vieweg, 2020.
ISBN: 978-3-658-30547-5.
DOI: 10.1007/978-3-658-30547-5.

[4] Christian Baun.
Betriebssysteme kompakt: Grundlagen, Daten, Speicher, Dateien, Prozesse und Kommunikation.
2. Auflage. IT kompakt.
Berlin und Heidelberg: Springer Berlin Heidelberg und Imprint: Springer Vieweg, 2020.
ISBN: 978-3-662-61411-2.
DOI: 10.1007/978-3-662-61411-2.

[5] Hongwei Zhou u. a. „LeMo: Protecting Kernel with Least Privilege Modules“.
In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, März 2019, S. 349–353. ISBN: 978-1-5386-6243-4.
DOI: 10.1109/ITNEC.2019.8729327.

[6] Hatem A. El-Azab u. a.
„Design and implementation of multicore support for a highly reliable self-repairing μ-kernel OS“.
In: *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*. IEEE, Dez. 2017, S. 13–22.
ISBN: 978-1-5386-0821-0.
DOI: 10.1109/INTELCIS.2017.8260021.

[7] Dongxu Ji u. a. „MicroTEE: Designing TEE OS Based on the Microkernel Architecture“.
In: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*.
IEEE, Aug. 2019, S. 26–33.
ISBN: 978-1-7281-2777-4.
DOI: 10.1109/TrustCom/BigDataSE.2019.00014.

[8] Simon Biggs, Damon Lee und Gernot Heiser.
„The Jury Is In: Monolithic OS Design Is Flawed: Microkernel-Based Designs Improve Security“. In: *Proceedings of the 9th Asia-Pacific Workshop on Systems*. APSys '18.
New York, NY, USA: Association for Computing Machinery, 2018.
ISBN: 9781450360067.
DOI: 10.1145/3265723.3265733.

[9] Xiaolong Wang u. a. „Enhanced Security of Building Automation Systems Through Microkernel-Based Controller Platforms“.
In: *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, Jun. 2017, S. 37–44.
ISBN: 978-1-5386-3292-5.
DOI: 10.1109/ICDCSW.2017.25.

[10] Nathan Dautenhahn u. a. „Nested Kernel: An Operating System Architecture for Intra-Kernel Privilege Separation“.
In: *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*.
ASPLOS '15. New York, NY, USA: Association for Computing Machinery, 2015, S. 191–206.
ISBN: 9781450328357.
DOI: 10.1145/2694344.2694386.

[11] Yuya Kobayashi, Masaya Sato und Hideo Taniguchi. „Evaluation of Processing Distribution for Application Program and OS in Microkernel OS“. In: *2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, Nov. 2019, S. 440–444. ISBN: 978-1-7281-5268-4.
DOI: 10.1109/CANDARW.2019.00083.

[12] Chen Zheng u. a. „XOS: An Application-Defined Operating System for Datacenter Computing“.
In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, Dez. 2018, S. 398–407.
ISBN: 978-1-5386-5035-6.
DOI: 10.1109/BigData.2018.8622507.

[13] Ahmad El-Rouby u. a.
„NileOS: A Distributed Asymmetric Core-Based Micro-Kernel for Big Data Processing“.
In: *IEEE Access* 9 (2021), S. 3696–3711.
DOI: 10.1109/ACCESS.2020.3048082.

[14] Matthias Hille u. a. „A heterogeneous microkernel OS for Rack-Scale systems“.
In: *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*.
Hrsg. von Taesoo Kim und Patrick P. C. Lee.
New York, NY, USA: ACM, Aug. 2020, S. 50–58. ISBN: 9781450380690.
DOI: 10.1145/3409963.3410487.

[15] Randolf Rotta u. a. „MyThOS — Scalable OS Design for Extremely Parallel Applications“.
In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*.
IEEE, Jul. 2016, S. 1165–1172.
ISBN: 978-1-5090-2771-2.
DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0179.