Operating Systems: Sheet 01

Feliks & Bennet

Aufgabe 1-1(Architekturen)

Vorhandene Betriebssystementwürfe können grob in zwei Klassen eingeteilt werden:

- Makrokernarchitektur
- Mikrokernarchitektur

Diskutieren Sie beide Architekturansätze unter Zuhilfenahme mindestens der folgenden Quellen:

- J. Liedtke, *Toward Real μ-Kernels*, *Communications of the ACM*, 39(9):70–77, September 1996. Verfügbar unter: https://citeseerx.ist.psu.edu/pdf/64f7ffba964343a3fc91f870672459b619988d6c
- C. Maeda, B.N. Bershad, *Networking Performance for Microkernels*, *Proceedings of the Third Workshop on Workstation Operating Systems*, 13:154–159, April 1992. Verfügbar unter: https://citeseerx.ist.psu.edu/pdf/4d5811e818391a007616b1ae881b1738d643e630

Lösung:

Der Kernel ist der Kern des Betriebssystems und stellt die grundlegende Infrastruktur bereit, die zwischen Hardware und Anwendungen vermittelt. Er bildet die privilegierte Softwareschicht, die die Hardware abstrahiert und die Grundkonzepte (Prozesse, Kommunikation) bereitstellt, auf denen das restliche Betriebssystem aufbaut.

Makrokernarchitektur (Monolithischer Kernel)

Bei dem Makrokernarchitektur werden alle Services innerhalb des Kernels implementiert. Dies kann Scheduling, Dateisystem, Networking, Gerätetreiber, Speichermanagement und Paging umfassen.

Vorteile

- Packete werden vom Kernel direkt an den Adressraum des Empfängers gesendet. Dadurch müssen Pakete nicht wiederholt Kopiert werden.
- Die Eingabeverarbeitung (Input Processing) kann auf der Interrupt-Ebene erfolgen, was die Latenz reduziert. Um die Latenz auf ein Minimum zu reduzieren, können Teile des Protokolls im Interrupt-Handler des Kernels platziert werden, wodurch ein zusätzlicher Dispatch und RPC entfallen.
- Bestimmte Services sind nicht angewiesen drauf Context Switches vorzunehmen, dadurch, dass sie auf Kernelebene platzieren sind

Nachteile

- Monolithische Syteme sind weniger modular
- Potenziell unisicherer dadurch, dass viele Services Zugriff auf den Kernel haben

Mikrokernarchitektur (μ-Kernel)

Die Mikrokernalarchitektur basiert auf der Idee den Kernel zu minimieren, deshalb werden mölichst viele Services auf die Nutzterebene verlagert. Der Kernel bietet nur essentielle Mechanismen wie Adressräume, Interprozesskommunikation (IPC) und grundlegendes Scheduling,

Vorteile

- Das OS wird flexibler und modularer, es lassen sich Hard- und Software leichter anpassen und austauschen
- Protokolle auf Benutzterebene lassen sich besser Anpassen und debugen als auf Kernelebene
- Eine Koexistenz verschiedener API's, File-Systeme oder sogar zusätzliche Betriebsysteme ist möglich.
- Jeder Server läuft in seinem eigenen Adressraum, wodurch die Komponenten voneinander geschützt sind. Eine Fehlfunktion eines Servers ist isoliert, ähnlich wie bei einer normalen Anwendung. Dies reduziert die TCB
- Durch die Schrumpfende Codebase gestaltet sich das wesentlich Wartbarer

Nachteile

- Programme übernehmen von exestierenden Macrokernelsystemen führ zu schlechter Performance dadurch, dass diese nicht der Art des Kernels angepasst sind
- Eines der Paper zeigt, dass nach Optimierung der Programme Microkernels eine ähnliche Performance zeigen aber kritisiert:
- Die Perfromance ist nicht wesentlich besser zu Macrokernels

Vergleich

Performance Die Papers befassen sich viel mit der Leistung der verschiedenen Architekturansätzte. Wie die Papers aufzeigte war die These das Das Mikrokernalarchitektur, gegenüber der Makrokernarchitektur deutlich Ineffizienzer sei, dies zeigenten auch Tests einzelner Netzwerkprotokolle, wo die Mikrokernalarchitektur 2 bis 5 mal länger brauchten. Dies konnte jedoch auf die falsche Implemnetierung und nicht auf die Architektur zurückgeführt werden.

Sicherheitsaspekte Die Mikrokernel-Architektur bietet klare Sicherheitsvorteile aufgrund der Kapselung.

- TCB-Reduktion: Die TCB (Trusted Computing Base) wird im Mikrokernel-Ansatz stark reduziert, da viele Dienste und Treiber nicht mehr als absolut vertrauenswürdig im Kernel laufen müssen.
- Fehlerisolation: Server-Fehlfunktionen sind isoliert. Ein Gerätetreiber kann beispielsweise nicht mehr Schaden anrichten als ein Editor.
- Makrokernel-Risiko: Im Makrokernel muss die gesamte Funktionalität innerhalb der TCB als vertrauenswürdig betrachtet werden. Ein Fehler könnte das ganze System zum Absturz bringen.