Architektur-Trends im Bereich Betriebssysteme

Perlen der Weisheit

M. Pizka, 07.08.01

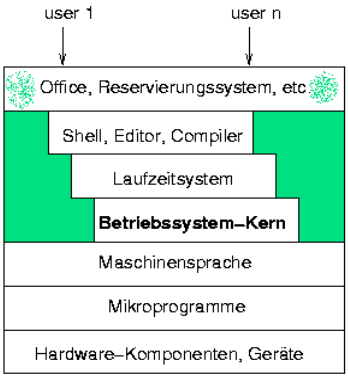
Überblick

- Einordnung und Abgrenzung
  - Betriebssystem
  - Architektur
- Separation: Evolution minimaler Kerne
- VBS-Fallstudie Distributed Shared Memory
- Integration: Flexibilisierung und Effizienz

Betriebssystem

- „Speicherresidentes, privilegiertes Programm unterhalb syscall Schnittstelle.“
- „Kern = BS“

- general purpose
- DIN 4430: „Das BS wird gebildet durch die P eines digitalen RS, die ... P steuern und überwachen.“



Automatisierung repetitiver und schwieriger Aufgaben.

3

Interessante BS-Eigenschaften

vielstufiger Übergang Modell → techn. Realisierung
- Speicher, Rechenfähigkeit, Zeit, ...
- vollständig automatisiert, optimiert

„Integration of (contradicting) concerns“
- Performanz,
- Security, ...

Wiederverwendung
- Adaptionsfähigkeit
- Langlebigkeit
- Kompatibilität

hohe Anforderungen an
- Funktionssicherheit
- Leistungsfähigkeit

4

Architektur - *buzzword*

Architekt:
- Grobentwurf, Detaillierung bis vollständige Beschreibung

Architektur i.d. Informatik:
- stark vergröberte Sicht, Verzicht auf Details
- Kriterien für Vergröberung?

Beobachtung:
- rekursiv
- multi-dimensional

BS komplexe Systemprogramme
- Linux: ca. 2 mloc
- welche Sicht (Gesamtsystem, C/S NFS-Architektur, ...)?

Terminologie: **BS-Strukturen**

5

BS-Trends

- Unterstützung großer Adressräume
- Security
- Multimedia, RT
- Mobile
- Miniaturisierung (Palm-OS, Embedded)
- Scalability+Portability
- Persistenz (Alternativen zu Dateien)
- **Parallele und Verteilte Systeme**
- **Tailoring, Customization und Extensibility**

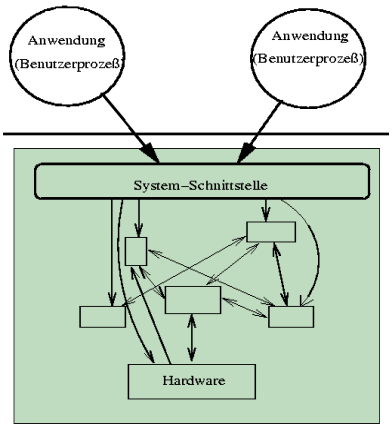
6

Monolithische Struktur

- Struktur: keine bis wenig
- Integration aller
  - Prozeduren
  - Daten

  in einen „großen“ Kern
- information hiding: nein
- Wartung: problematisch

✓ kurze Pfade, performant
✓ am weitesten verbreitet



7

BSD 4.4 Kern

<table>
<tr><td colspan="4">System calls</td><td colspan="3">Interrupts and Traps</td></tr>
<tr><td colspan="2">Terminal handling</td><td>Sockets</td><td>File naming</td><td>Map ping</td><td>Page faults</td><td rowspan="2">Signal handling</td><td rowspan="2">Process creation and termination</td></tr>
<tr><td rowspan="2">Sys calls</td><td>Cooked tty</td><td>Network protocols</td><td>File systems</td><td colspan="2">Virtual memory</td></tr>
<tr><td>Line discipl.</td><td>Routing</td><td>Buffer cache</td><td colspan="2">Page ceache</td><td colspan="2">Process scheduling</td></tr>
<tr><td colspan="2">Character devices</td><td>Network device drivers</td><td colspan="2">Disk device drivers</td><td colspan="2">Process dispatching</td></tr>
<tr><td colspan="7">Hardware</td></tr>
</table>

8

Schichtung – The (Dijkstra 68)

- Reduktion der Komplexität der Abhängigkeiten
- Abhängigkeiten von Schicht S(n+1), S(n), S(n-1)
- Horizontale Struktur

- Vertikale Struktur fehlt
- Anzahl der Schichten?
- Erhöhte Komplexität durch Vermittlerkomponente (Linearisierung)
- Effizienzverluste

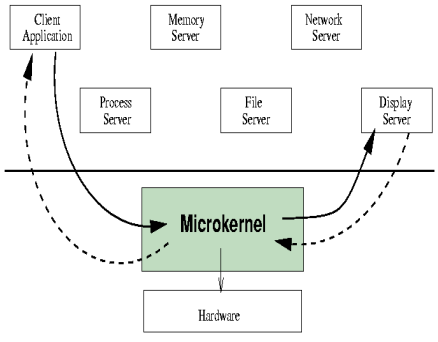
Schicht 5	Benutzerprogramme
Schicht 4	Virtuelle E/A-Geräte
Schicht 3	Virtuelle Operator-Konsole
Schicht 2	Speicherverwaltung
Schicht 1	Prozeßverwaltung
Schicht 0	Hardware

Verallgemeinerung: MULTICS-Ringe

9

Mikrokern

- Mach (83), Chorus, Amoeba
- Mach: multiprocessors
- matching application neeeds
  - minimaler Kern
  - Dienste in Server-Prozessen
  - dynamisch erweiterbar
- Trennung Strategie Mechanismus
- OS Personalities (DOS/UNIX)
- IPC: explizit, n.-orientiert
- Kriterium für Kernintegration?

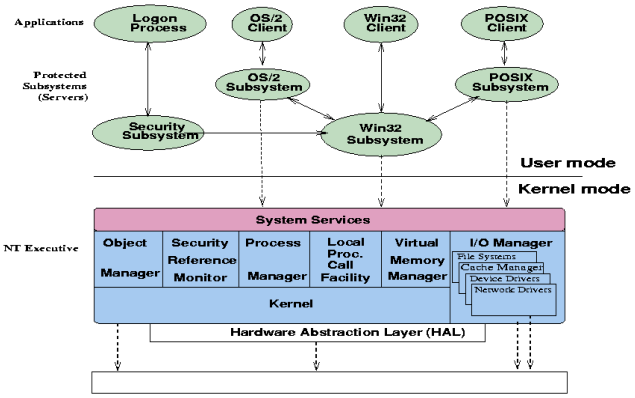


Linus Torvalds:
„Linux is a µ-kernel OS. The kernel only contains what's necessary“

10

Windows NT/2000

(29mloc)



kein µ-Kern basiertes System

Erweiterbare Kerne

- Beobachtung existierender µ-Kerne:
  - ineffizient (Konflikt mit GP-Flexibilität)
  - schwer zu warten, geringe Wiederverwendung
- Analyse
  - µ-Kerne sind nicht µ (mem+interface)
  - µ-Kerne sind nach wie vor Monolithen
  - µ-Kerne sind nicht adaptierbar
- Lösungsansatz: dynamisch erweiterbare Kerne
  - optimale Nutzung von Ressourcen durch
  - anwendungsspezifische Anpassung

Bsp.: SPIN, Aegis ExOS, Cache Kernel, Apertos, Scout

Ansätze

SPIN, Cache Kernel – extensible kernel
- SPIN: user-defined spindles
- CK: Kern cached Threads und Adressräume als benutzermodellierte Anwendungskerne

Apertos – reflective meta-abstraction
- Meta-Abstraktionen für die Konstruktion von Abstraktionen
- Anwendungen spezifizieren eigene Abstraktionen (not fixed!) µ
- Komposition individueller Ausführungsumgebungen

Aegis, Panda, L4 – no (minimal) abstractions
- Abstraktionen sind die Wurzel aller Ineffizienzen
- keine Kernabstraktionen
- Anwendungen konstruieren Abstraktionen selbst

13

Picokerne – L4 (J. Liedtke, ca.95)

- „outside µk whatever possible“
  - Modularität, Isolation koexistierende Strategien
  - L4: VA, Threads, IPC, UID
- Mach-Ineffizienzen
  - k-u switch: 900 statt 107c
  - VA switch: 800 statt 50c
  - Thr switch: 600µs/10µs
  - Ursachen: Implementierung
- L4-Ergebnisse:
  - µ-Kerne sind nicht portabel
  - Implementierung muß Prozessorspezifika nutzen (486, PPro, PowerPC)

	486	Pentium
TLB	32 entries 4 ways	32+64 entries 4 ways
Cache	8K 4ways 16-bye lines write through	8K+8K 2ways 32-byte lines write back
SegReg trap	9 cycles 107 cycles	3 cycles 69 cycle

14

Synthesis und Synthetics

- hoch-optimierte, nicht-portable Kerne
- dynamische, inkrementelle Spezialisierung
  - Interpretation → Transformation
  - partielle Auswertung von Invarianten
  - Spekulation über Quasi-Invarianten
- Realisierung: Kern-integrierter Übersetzer
  - generiert Kern-Code zur Laufzeit
  - Beispiel: optimiertes Lesen aus Pipe
- Probleme:
  - Schwellwerte
  - auf modernen Architekturen Konflikt mit Caches

15

... und das Ende der μ-mania?

- P. Druschel, V. Pai and W. Zwaenepoel (98)
  „Extensible Kernels are Leading OS Research Astray“
  - ... only performance related results so far
  - ... raise difficult compatibility and safety questions
  - ... try to solve the safety problems they introduced
  - ... app-specific kernel-extensions not required
  - ... fails to address real OS challenges
- pragmatic approach
  - extensions only to be used in experimental settings
  - after development tightly integrated into base OS
  - let OS community refocus on real challenges (e.g. HTTP)

erweiterbare Monolithen ausreichend (LynxOS, Linux)

16

Verteilte Betriebssysteme

- VBS ≠ Netzwerkbetriebssystem
  - Transparenz der Verteilung
  - u.a. keine Eigentümerschaft
- zahlreiche Forschungsarbeiten (90er)
  - Comandos (EU), Amoeba (Tan), Mach (CMU),..., MoDiS (TUM)
  - schwach integrierte Ergänzungen: CORBA, PVM, MPI, DCE
  - kommerzielle VBS?
- ungelöste Probleme
  - w.A. (Lamport/Agrawala unrealistisch)
  - garbage collection (weder refcounting noch mark'n sweep)
  - deadlock-detection, ...
- unbefriedigende Leistungsfähigkeit

17

Problem Leistungsfähigkeit

- local/remote Zugriff: 10ns/2ms ≈ $5*10^5$
- entfernte Zugriffe kritisch ⇒ minimieren
- Strategie anwendungsabhängig
- GP-VBS: hohe Flexibilität erforderlich
- Flexibilität verursacht a priori Mehraufwand
- Bsp: Orca/Amoeba – Matrixmult, Zeiten in Sek.

P	1	2	3	4	5
Orca	810.3	410.6	279.1	209.8	169.9
SMM	180.5	90.5	60.8	45.9	36.6

lineare Beschleunigung, aber inakzeptable Effizienz

18

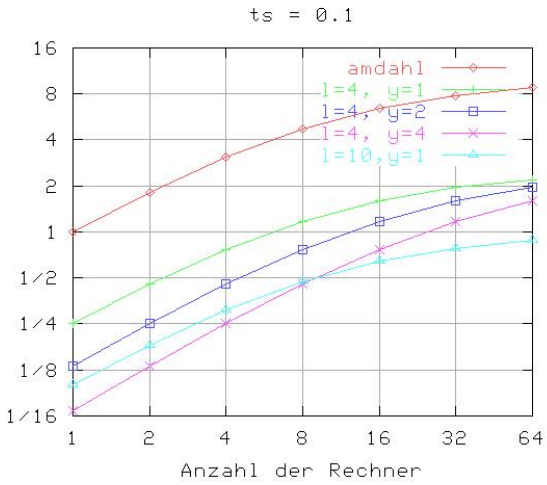
Leistung in verteilten Systemen

Amdahl

$$s_r = \frac{t_s + t_p}{t + t_p / n} = \frac{n}{1 + (n-1) * t_s}$$

verteilt

$$S_v(y,l) = \frac{n}{(t_s * (n-y) + y) * l}$$

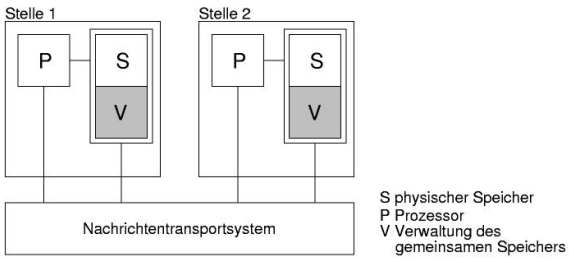


19

Fallstudie DSM

Distributed Shared Memory

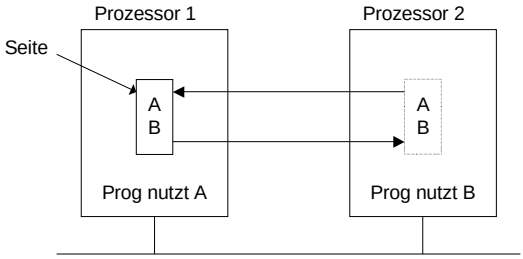
- bisher: message-passing (MPI, PVM)
- Idee: shared-memory Paradigma von Multiprozessorsystemen in Software



20

Paged-Based DSM

- Bsp.: Ivy (Li´86), JiaJia, Quarks, Murks (MP´00), ...
- Transport-Einheit: HW-Seite (4kB, 8kB)
  - HW-Support (MMU): Seitenfehler
  - prefetching Charakteristik
  - false sharing – inakzeptable Leistung



21

Lösungsansätze

- Partitionierung der Daten durch Programmierer
  - negativer Effekt auf Caching
  - Verlust von: Transparenz, Einfachheit, Skalierbarkeit
- O-basierte DSM (Linda, Munin)
  - Einheit: Anwendungsobjekte / Variablen
  - kein MMU-Support ⇒ mem read/write über Bibliothek
  - was sind geeignete Objekt? Bsp: int a[4096];
- Abschwächung der Konsistenzforderung
  - sequential, causal, weak, release, entry, ...
  - kompliziertes Programmiermodell

22

DSM Integration – Compiler

- Treadmarks, Orca,...
  kombinierte Compile-/Run-Time DSM-Systeme
  - automatische Partitionierung der Daten
  - präzise Information über Daten- und Kontrollfluß
  - statische Code-Transformation
- komplexe Kopplung:
  - BS-Kern
  - Laufzeitsystem
  - Übersetzer (Binder)

23

Der Monolith ist tot, es lebe der Monolith

- steigende Vielfalt von
  - Anforderungen und
  - Angeboten
- häufig keine uniforme, optimale Lösung
- Effizienz benötigt Flexibilität
- Flexibilität benötigt Integration
  - Information
  - Realisierungsalternativen

... wie werden geeignete Strategien gefunden?

24