Betriebssystemarchitektur von MEMSY

Thomas Thiel

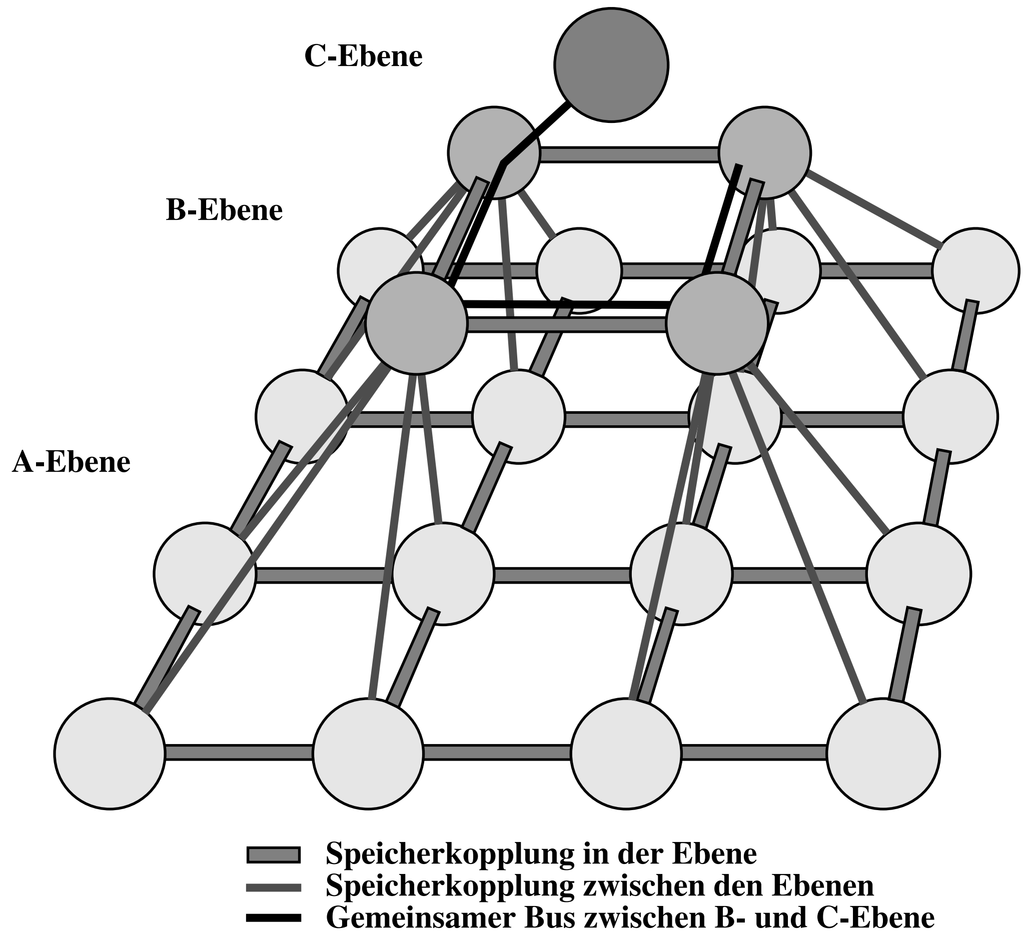
Universität Erlangen-Nürnberg — IMMD IV
SFB 182 TP B2
thiel@informatik.uni-erlangen.de

Überblick

- ❑ Einführung - Experimentalsystem MEMSY
- ❑ Betriebssystemarchitektur
  - ➔ UNIX
  - ➔ eigene Erweiterungen
- ❑ Zusammenfassung und Ausblick

Experimentalsystem MEMSY

Einführung



Betriebssystem

- ❑ Auf jedem Knoten eigener UNIX-Betriebssystemkern
  - ➔ Mehrbenutzerbetrieb
  - ➔ Mehrprogrammbetrieb
  - ➔ UNIX-Programmierumgebung
- ❑ Betriebssystem mit notwendigen eigenen Erweiterungen
  - ➔ Anpassung an die vorhandene Hardware
    - Kommunikationsspeicher
    - Interrupts zwischen den Knoten
    - Debug-Hardware
  - ➔ Standard UNIX Anwendungen ausführbar
- ❑ Unterstützung des Programmiermodells

UNIX SYSTEM V/88 Release 3

Betriebssystem

❑ Multiprozessorbetriebssystem mit *Peer-Processors* Architektur

❑ Parallelismus auf Prozeßebene

➜ UNIX Prozesse sind unverändert auf MP-Architektur ausführbar

❑ Zwei-Teilung des Kerns

❶ Voll parallel ausführbare Bereiche bzw.
feingranulares Locking für gegenseitigen Ausschluß

❷ Ein großer Bereich durch einen *Kernel-Semaphor* geschützt

❑ Höherer Systemdurchsatz durch

➜ parallelen Zugriff auf gepufferte E/A

➜ voll parallele Ausführung, wenn außerhalb des Kerns

Betriebssystemerweiterungen

- ❑ Anwendungsunterstützung
  - ➔ Einführung von Applikationen
  - ➔ Veränderung des Prozessorvergabemechanismus
- ❑ Kommunikationsspeicher
  - ➔ Aufteilung
  - ➔ Verwaltung
- ❑ Kommunikation und Koordinierung
  - ➔ Unterbrechungen
  - ➔ Nachrichten
  - ➔ Shared-Memory
  - ➔ Semaphore
  - ➔ Signale

Anwendungsunterstützung

Betriebssystemerweiterungen

❑ Anwendungsunterstützung durch Applikationskonzept

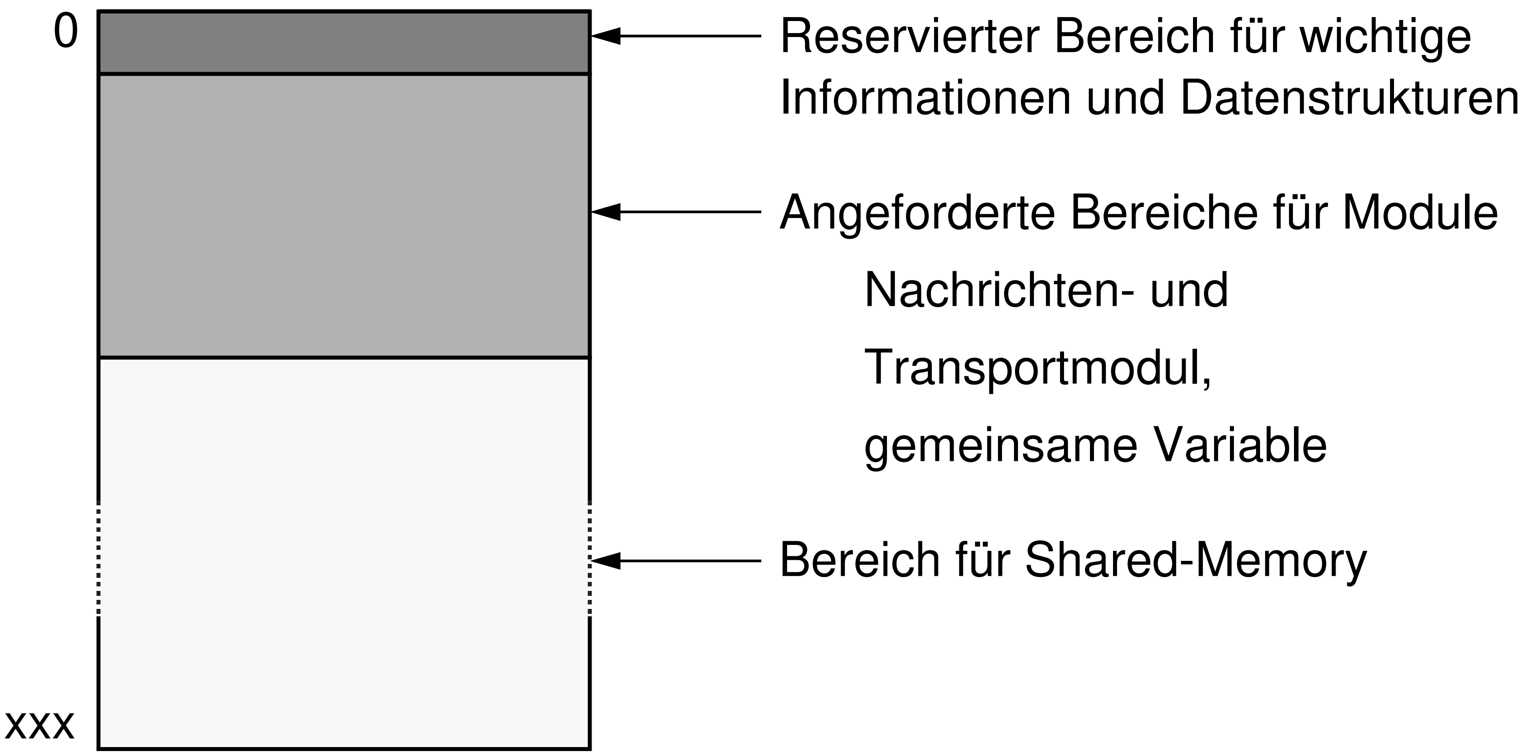
- Alle Prozesse einer Anwendung werden zu einer Applikation zusammengefaßt (knotenübergreifend)
- Eine Applikation wird durch eine global eindeutige Applikationsnummer identifiziert
- Prozesse einer Applikation werden durch Knoten-ID, Applikationsnummer und Jobnummer identifiziert
- Mehrere voneinander unabhängige Applikationen sind auf dem Gesamtsystem gleichzeitig möglich
- Eine ausgezeichnete Applikation wird bereits beim Systemstart erzeugt
- Ein Applikations-Dämon pro Knoten überwacht und regelt das Erzeugen und Zerstören von Applikationen

Prozessorvergabe

- ❑ Erweiterung des SYSTEM V Prozessorvergabemechanismus durch zusätzliche *Applikationswarteschlangen*
  - Applikationsprozesse werden ausschließlich in der zugehörigen Applikationswarteschlange eingetragen
  - Alle sonstigen Prozesse werden in der *Systemwarteschlange* eingetragen
  - Prozesse mit Systempriorität werden vor allen anderen Prozessen bedient
  - Applikationen wird (analog zu Prozessen) eine Priorität zugewiesen
  - Prioritäten von Applikationen werden dynamisch durch einen Serverprozeß verändert
  - Lauffähige Prozesse der Applikation mit der höchsten Applikationspriorität werden bevorzugt

Kommunikationsspeicher

- ❑ Aufteilung der Kommunikationsspeicher
  - Jedem Knoten ist ein Kommunikationsspeicher zugeordnet
  - Alle Kommunikationsspeicher sind gleich aufgebaut



Kommunikationsspeicher

Betriebssystemerweiterungen

❑ Verwaltung der Kommunikationsspeicher (auf Kernebene)

- Einfache Schnittstelle für Kernmodule:
  - Reservieren von zusammenhängendem Speicher
- Einfache Schnittstelle für den Shared-Memory-Mechanismus:
  - Anfordern von gemeinsamen Speichersegmenten
  - Freigabe von bereits angeforderten Segmenten
  - Einblenden von Speichersegmenten in den virtuellen Kernadreßraum
  - Ausblenden aus dem virtuellen Kernadreßraum
- Verwaltung über Tabellen im Kommunikationsspeicher

Kommunikation

Betriebssystemerweiterungen

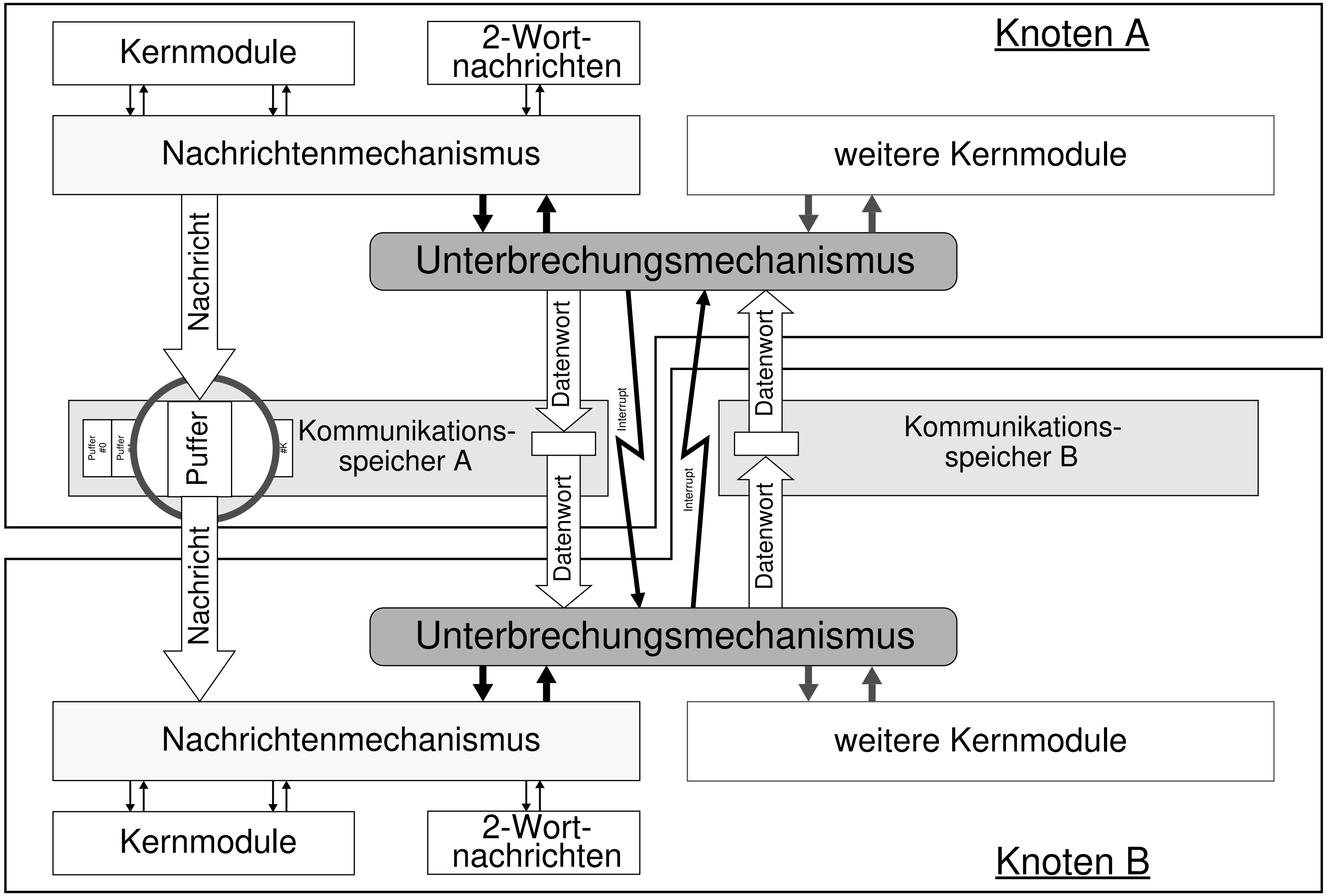
❏ Nachrichtenmechanismus

- Einfache Schnittstelle für alle Kernmodule
- Sendemechanismus blockierend und nicht-blockierend
- Empfangsmechanismus sorgt für Pufferung und Verteilung der Nachrichten
- Überwachung der Ressource-Konsistenz (z. B. bei Knotenausfällen)
- Weitergabe von Zustandsinformationen an die Kernmodule

Nachrichtenmechanismus

Betriebssystemerweiterungen
Kommunikation

❏ Senden einer Nachricht von Knoten A zu Knoten B



Kommunikation

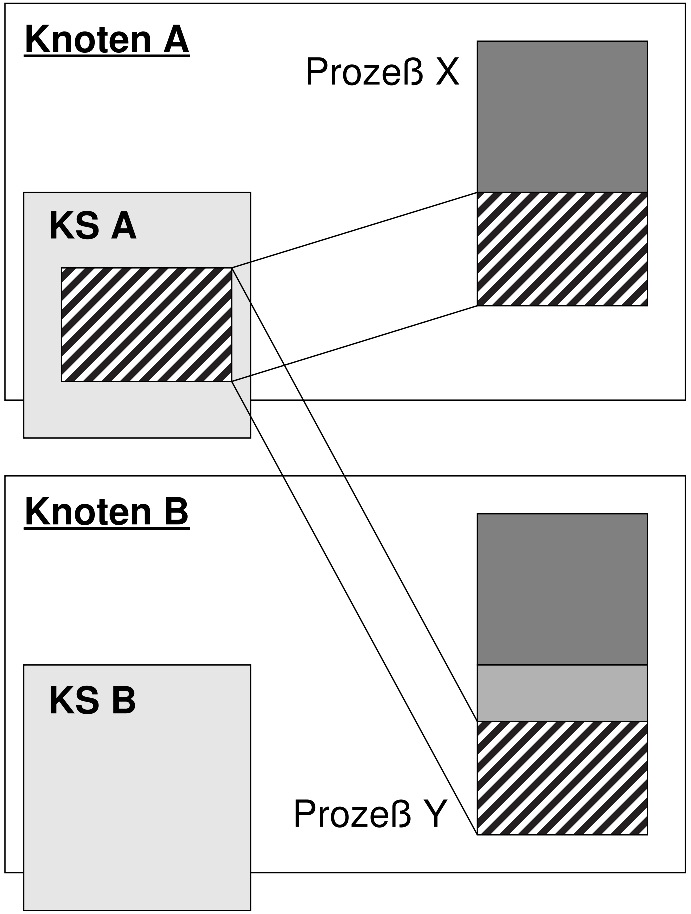
Betriebssystemerweiterungen

❑ Shared-Memory

- Erzeugen von gemeinsamen Speichersegment nur im eigenen Kommunikationsspeicher möglich

- Vier Möglichkeiten für das Zerstören von gemeinsamen Segmenten
  - Zerstören des Speichersegments, wenn der Besitzer es ausblendet
  - Zerstören, wenn es der letzte referenzierende Prozeß ausblendet
  - Zerstören, wenn die Applikation terminiert
  - explizit

- Kommunikation mit Nachbarn über den Nachrichtenmechanismus



Zusammenfassung und Ausblick

Erfahrungen

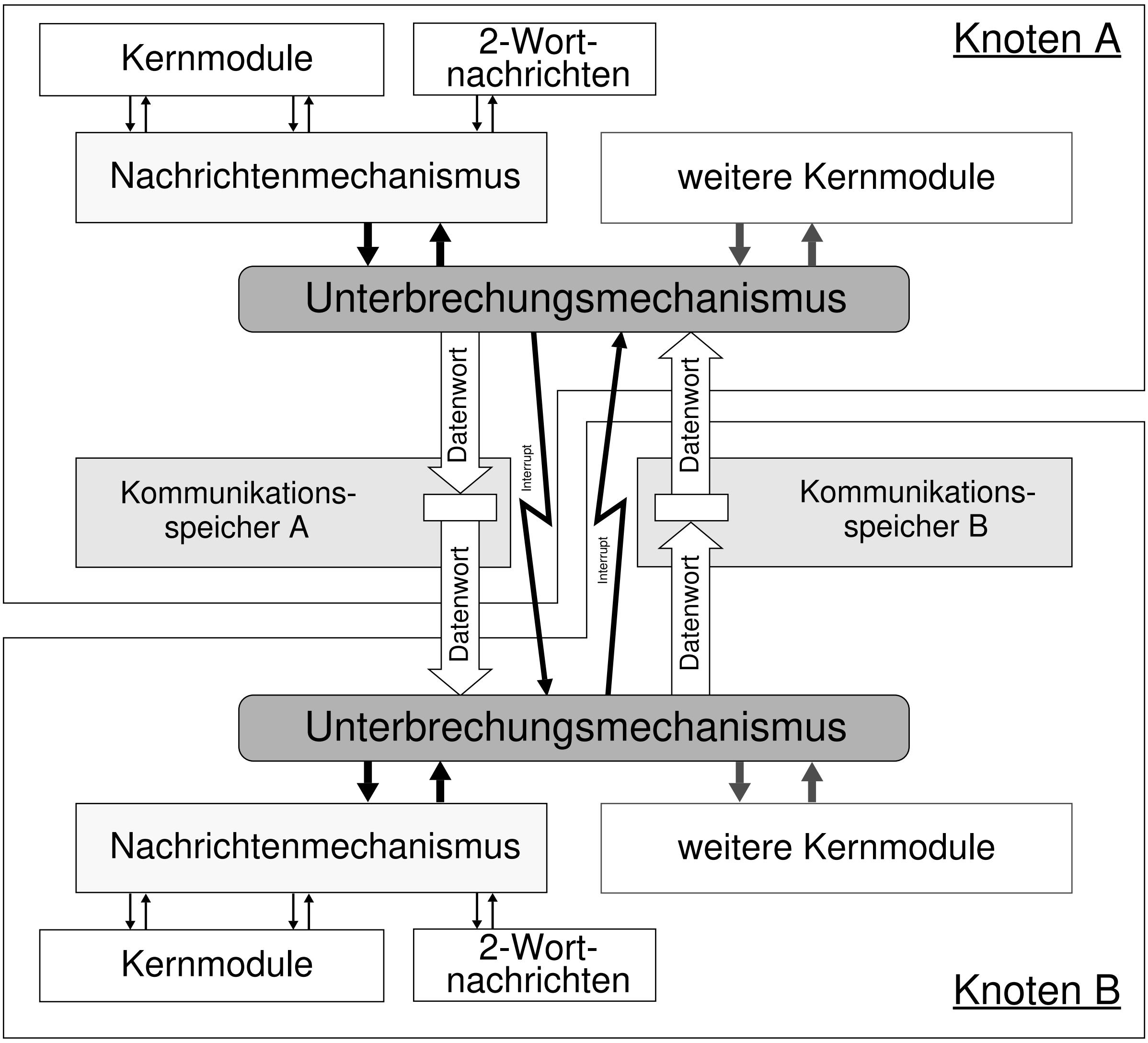
- ❑ Kurze Entwicklungszeit durch Verwendung eines existiernden Betriebssystems
- ❑ Anwendungsunterstützung durch die Abstraktion der Applikation
- ❑ Auswahl an verschiedenen Kommunikationsmechanismen

Weitere Arbeiten

- ❑ Ausdehnung der Kommunikations- und Koordinierungsmechanismen auf das gesamte System
- ❑ Aufheben der physikalischen Beschränkung des Kommunikationsspeichers durch Auslagern von gemeinsamen Speichersegmenten

❏ Nachrichtenmechanismus

- Schnittstelle für alle Kernmodule
- Blockierender / nicht-blockierender Sendemechanismus
- Pufferung und Verteilung von empfangenen Nachrichten
- Überwachung der Ressource-Konsistenz
- Statusmeldungen



Kommunikationsspeicher

- Verwaltung der Kommunikationsspeicher (auf Kernebene)