

Betriebssystemtechnik

Adressräume: *Trennung, Zugriff, Schutz*

XII. Nachlese

Wolfgang Schröder-Preikschat

18. Juli 2019

Gliederung

Rekapitulation
Prozessadressräume

Perspektiven
Forschungsschwerpunkte und -projekte
Rechnerausstattung
Weiterqualifikation



© wosch

BST (SS 2019, VL 12)

Rekapitulation

2

Adressraumverwaltung

querschneidender Belang



Verfeinerung/Vertiefung von Betriebssysteme

[1, 2]

Adressräume (von Programmen/Prozessen)

- **trennen**: in eine räumliche Distanz voneinander bringen
 - klassisch, hardwarebasiert, durch MMU *und* Betriebssystem
 - unterstützt durch Dienstprogramme (*utility program*)
 - Kompilierer, Assemblierer, Binder, Lader
 - vertikal (vom Betriebssystem) und horizontal (Anwendungsprogramme)
- **zutreffen**: nach etwas greifen und es festhalten bzw. an sich nehmen
 - Interprozesskommunikation (IPC) und Mitbenutzung (*sharing*)
 - Mitbenutzung durch Daten- (*data*) und Textverbund (*code sharing*)
 - kopieren beim Schreiben/Referenzieren (*copy on write/reference*)
- **schützen**: einer Sache Schutz gewähren, einen Schutz [ver]schaffen
 - Angriffssicherheit (*security*) und Betriebssicherheit (*safety*)
 - Immunität einerseits und Isolation andererseits
 - Eindring bzw. Ausbruch von Prozessen verhindern
- ergänzend: softwarebasiert, durch typsichere Programmiersprachen



© wosch

BST (SS 2019, VL 12)

Rekapitulation – Prozessadressräume

4

Rekapitulation Prozessadressräume

Perspektiven Forschungsschwerpunkte und -projekte Rechnerausstattung Weiterqualifikation



Fehler- und Einbruchstoleranz

Betriebsmittel schonende **byzantinische Fehlertoleranz** (BFT)

- **Virtualisierung** als Schlüsseltechnologie
 - Konsolidierung von Diensteeinheiten (*server*)
 - Redundanz durch replizierte virtuelle Maschinen
- **Vorhersage** wahrscheinlicher Ausführungspfade
 - deterministische mehrfädige Ausführung
 - geordnete Sperrreihenfolge (*lock sequence*)
 - Koordinierung zwischen Kopien als Ausnahmefall
 - skalierbare Sperrüberwachung und -verwaltung
- **minimal invasive Operation** für den Normalfall: $f + 1$ Kopien
 - dehnfähige (*resilient*) Einigungsprotokolle für den Ausnahmefall
 - performante Nachrichtenauthentifikation und -verifikation
 - zuverlässiger hardwarebasierter (FPGA) Zählerzuweisungsdienst
- DFG: seit 10/2009, 2 WM (1 FAU, 1 TUBS), 2 SHK



Forschungsschwerpunkte am Lehrstuhl

- **Komponierbarkeit und Konfigurierbarkeit**
 - anwendungsorientierte (variantenreiche, typsichere) Systemsoftware
 - **Sparsamkeit**
 - ressourcen-gewahrer Betrieb von Rechensystemen
 - **Zuverlässigkeit**
 - Betriebsmittel schonende Fehler- und Einbruchstoleranz
 - **Rechtzeitigkeit**
 - Migrationspfade zwischen zeit- und ereignisgesteuerten Echtzeitsystemen
 - **Spezialisierbarkeit**
 - dedizierte Betriebssysteme: integriert, adaptiv, parallel
 - **Gleichzeitigkeit**
 - Koordination der Kooperation und Konkurrenz zwischen Prozessen
- ↳ Prozessadressräume sind mehr oder weniger querschneidend dazu



Adaptive responsive eingebettete Systeme

Bereitstellung selektiver und adaptiver Fehlertoleranzmaßnahmen

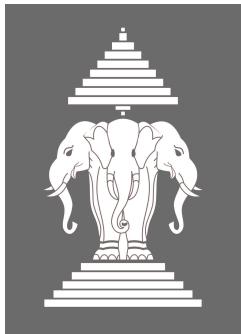
- **kombinierter Redundanzansatz**
 - holistische Absicherung v. Regelungsanwendungen
 - bestehend aus Sensorik, Regelung und Aktuatorik
 - der sicherheitskritische Teil des Gesamtsystems
 - softwarebasierte Fehlertoleranz
 - transiente Hardwarefehler tolerieren
- auf **gemischte Kritikalität** (*mixed criticality*) ausgerichtet
 - Echtzeitfähigkeit (weich, fest, hart) eng gekoppelt mit
 - Betriebs- (*safety*) und Angriffsicherheit (*security*)
- **Messunsicherheit** refl. Entwurfskonzept für Regelungssysteme
 - problemspezifische Modularisierung von Regelungsanwendungen
 - Aufbau zuverlässiger System aus unzuverlässigen Hardwarekomponenten
- BayStM WIVT: seit 01/2010, 2 WM, 2 SHK



Latenzgewährheit in Betriebssystemen für massiv-parallele CPUs

■ Latenzvorbeugung

- domänen spezifische Entwurfsmuster
- sperr- und wartefreie Synchronisation



■ Latenzvermeidung

- Interferenzschutz
- Eindämmung von Wettstreitigkeiten

■ Latenzverbergung

- asynchrone Prozedurfern-/Systemaufrufe
- Kerne für Betriebssysteme nutzen

■ Experimente mit unterschiedlichen Betriebssystemarchitekturen

- Entwicklung eigener prozess- und ereignisbasierte Betriebssystemkerne
- Übertragung mancher Konzepte und Techniken in Linux

■ DFG: seit 05/2011, 2 WM, 2 SHK

³<http://univis.uni-erlangen.de> → Forschungsprojekte → LAOS

■ **Octo** — der Bezeichnung eines Wesens entnommen, das:

- hoch parallel in seinen Aktionen ist und
- sich sehr gut an seine Umgebung anpassen kann

→ der Krake (Ordnung *Octopoda*)

- kann kraft seiner (acht) Tentakel parallel agieren
- vermag sich durch Farbänderung anzupassen und
- verfügt über ein hoch entwickeltes Nevensystem
 - um sich auf dynamische Umgebungsbedingungen und -einflüsse einzustellen



■ **POS** — Abk. für (engl.) *Parallel Operating System*

- ein Betriebssystem, das nicht bloß parallele Prozesse unterstützt
- sondern dabei selbst **inhärent parallel** arbeitet
- sowie sich einem wechselnden Anwendungsprofil entsprechend anpasst
 - Adressraumvirtualisierung und -devirtualisierung zur Laufzeit bei Bedarf
 - einhergehend mit dem Auf- und Rückbau des Betriebssystemkerns im Betrieb

■ DFG: seit 06/2011, 3.5 WM (2.5 FAU, 1 KIT), 1 WHK, 3 SHK

⁴<http://univis.uni-erlangen.de> → Forschungsprojekte → iRTSS

Aspektorientierte Echtzeitsystemarchitekturen

Reflektion kausal und temporal abhängiger gleichzeitiger Aufgaben

■ zeitgesteuerte (time-triggered, TT) Systeme

- Vorabwissen zwingend erforderlich
- statische Ablaufplanung, vor Laufzeit
- implizit koordinierte Prozesse, kein Laufzeitaufwand
- von eher simpler Struktur, leichter analysierbar



■ ereignisgesteuerte (event-triggered, ET) Systeme

- Vorabwissen nicht erforderlich, aber vorteilhaft
- dynamische Ablaufplanung, zur Laufzeit
- explizit zu koordinierende Prozesse, Laufzeitaufwand
- von eher komplexer Struktur, schwieriger analysierbar

■ Migrationspfad zwischen verschiedenen Echtzeitsystemarchitekturen

- übersetzungsgestützte Transformation von ET- zu TT-Programmen v.v.
- kanonische Programmierschnittstelle für Echtzeitanwendungssysteme

■ DFG: seit 08/2011, 2 WM, 2 SHK

⁵<http://univis.uni-erlangen.de> → Forschungsprojekte → AORTA

Softwareinfrastruktur vernetzter Sensorsysteme

Adaptive Run-Time Environment betriebsmittelarmer Sensorsysteme

■ mobiles (agiles) Sensornetz einerseits

- Mausohr *Myotis myotis* mit „Sensorknotenrucksack“
 - Ortswechsel, Migration und Soziobiologie
- Fliegengewicht (20 g), fordert extrem leichten Aufbau
 - Kleinstknoten (2 g) → wenig Hardware, hochintegriert
- Systemsoftware für winzigste Rechensysteme
 - extrem wenig Speicher und Rechenleistung
 - dynamisches Laden bei *ad-hoc* Netzwerkverbindungen
 - Betriebsdauer maximieren, Energieverbrauch minimieren
- statische Programmanalyse zur Energiebedarfsabschätzung



■ stationäres Sensornetz andererseits: Netzkoppler (gateway)

- spontane und kurzzeitige Bewirtung der vorbeifliegenden Sensorknoten

■ DFG: seit 07/2012, 1 WM (0.5 FAU, 0.5 TUBS), 1 SHK

⁶<http://univis.uni-erlangen.de> → Forschungsprojekte → BATS

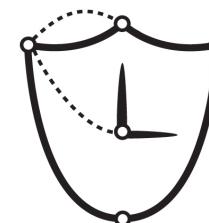
- Softwarekontrollierte Konsistenz/Kohärenz für vielkernige Prozessoren
- **ereignisbasierter Minimalkern**
 - zwischen Speichergerwahrer Speicherabdruck
 - „überfaden“ (*hyper-threading*) latenter Aktionen
- **federgewichtige Einigungsprotokolle**
 - kernübergreifende Synchronisation
 - Familie von Konsistenzkernen
- **problemorientierte Konsistenzmaschinen**
 - sequentielle, Eintritts- und Freigabekonsistenz
 - funktionale Hierarchie von Konsistenzdomänen
 - Speicherdomänen für NUMA-Architekturen
- Auslegungen für unterschiedliche **Prozessorarchitekturen**
 - partiell bzw. total, {in,}kohärenter gemeinsamer Speicher
- DFG: seit 08/2012, 2 WM (1 FAU, 1 BTU)



⁷<http://univis.uni-erlangen.de> → Forschungsprojekte → COKE

Latenz- und Elastizitätsbewusster Netzwerkbetrieb

- Latency- and Resilience-Aware Networking*
- **echtzeitfähige Netzwerkkommunikation**
 - Transportkanäle für Cyber-physische Systeme
 - vorhersagbare Übertragungslatenz
 - in gewissen Grenzen garantierte Gütemerkmale
 - **deterministische Laufzeitunterstützung**
 - Auffassung von der kausalen [Vor]bestimmtheit allen Geschehens bzw. Handelns (Duden)*
 - latenzbewusste Kommunikationsendpunkte, optimierter Protokollstapel
 - spezialisierte Ressourcenverwaltung, vorhersagbares Laufzeitverhalten
 - in zeitlichen (Phase 1) und energetischen (Phase 2) Belangen
 - DFG: ab 09/2016, 2 WM, 2 SHK (je 1 FAU, 1 Uni Saarland)



⁹<http://univis.uni-erlangen.de> → Forschungsprojekte → LARN

- **skalierbare Synchronisation durch agile kritische Abschnitte**
 - Infrastruktur ■ lastabhängiger und selbstorganisierter Wechsel des Schutzes vor Wettlaufsituationen
 - Sprachunterstützung ■ Vorbereitung, Charakterisierung und Erfassung deklarierter kritischer Abschnitte
- **automatisierte Extraktion kritischer Abschnitte**
 - Beschreibungssprache für kritische Abschnitte
 - Programmanalyse und LLVM Integration/Adaption
- **energie-gewahre Systemprogrammierung**
 - wechselseitiger Ausschluss, überwachter Abschnitt, Transaktion
 - dynamisches Binden von Schutzprotokollen bzw. kritischen Abschnitten.
- **manipulationssichere Energieverbrauchsmessung**
 - Befehlsübersicht und -statistik mit realen und virtuellen Maschinen
 - Vorhersage bzw. Abschätzung des Energieverbrauchs
- DFG: seit 01/2015, 2 WM, 2 SHK



⁸<http://univis.uni-erlangen.de> → Forschungsprojekte → PAX

Systeme mehr-/vielkerniger Prozessoren

faui4*	clock	cores per domain		domain		Xeon
		physical	logical	NUMA	tile	
8e	2.9 GHz	8	16	2	–	Opteron
8f	–	–	–	–	–	Xeon
9big01	2.5 GHz	6	–	8	–	Xeon
9big02	2.2 GHz	10	20	4	–	Xeon
9phi01	1.2 GHz	6	12	2	–	Xeon Phi
	1.1 GHz	57	228	2	–	
scc	1.5 GHz	4	2	1	–	Xeon
	800 MHz	2	–	–	24	Pentium
InvasIC	3.5 GHz	8	16	2	–	Xeon
	25 MHz	4	–	–	6	LEON/SPARC

- **budgeted acquisition:** weitere *n*-kernsysteme, transaktionaler Speicher
- **OctoPOS** ■ $n \geq 64$, in 2015
- **PAX** ■ $n \geq 16$, in 2016, zusätzlich mehrkernige Mikrocontroller



- [1] LOHMANN, D. ; SCHRÖDER-PREIKSCHAT, W. :
Betriebssysteme.
http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_BS, 2008 ff.
- [2] SCHRÖDER-PREIKSCHAT, W. ; KLEINÖDER, J. :
Systemprogrammierung.
http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_SP, 2008 ff.

