Untersuchung zum Kommunikations- und Debugverhalten von parallel koexistenten Betriebssytemen

Vorstellung der Diplomarbeit

Ulf Wetzker
Matrikelnummer: 35218
ulf.wetzker@stud.tu-ilmenau.de

Fachgebiet Integrierte Hard- und Softwaresysteme
Technische Universität Ilmenau

29. September 2008

Übersicht

Aufgabenstellung

Einleitung

Analyse und Lösungsansätze

Zusammenfassung

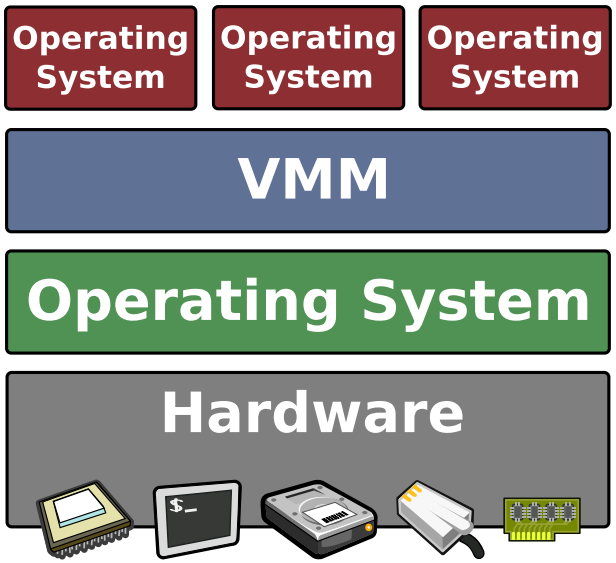
Ausblick

Aufgabenstellung

1. Literaturstudium zur Arbeitsweise des SMP-Linux-Kernels und des Debuggers UDE
2. Analyse und Konzeptentwicklung zur ausschließlichen oder gemeinsamen Nutzung von Betriebsmitteln
3. Entwurf eines Ansatzes zur betriebssystemübergreifenden Kommunikation
4. Vorschläge für neue Systemsichten und Abstraktionen bei der Fehlersuche in parallel koexistenten Betriebssystemen
5. Auswahl oder Realisierung geeigneter Benchmarks zur Leistungsbewertung
6. Demonstration, Dokumentation und Bewertung der realisierten Funktionen

Multi-Core-Prozessoren in eingebetteten Systemen

- Verbesserung des Rechenleistung–Leistungsaufnahme–Verhältnis
- Reduktion der Baugröße durch Verwendung gemeinsamer Komponenten und mehrerer Prozessorkerne in einem Gehäuse



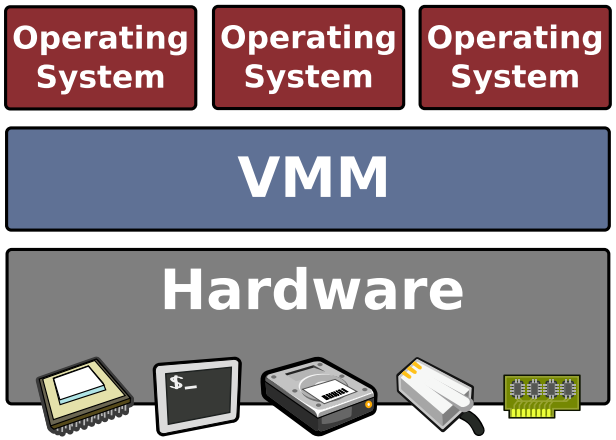
Virtual machine monitor
Type 2

Beispielanwendung

- Ausführung mehrerer Betriebssysteme auf einer Multi-Core-Plattform
- Mehraufwand bei Emulation oder Virtualisierung

Multi-Core-Prozessoren in eingebetteten Systemen

- Verbesserung des Rechenleistung–Leistungsaufnahme–Verhältnis
- Reduktion der Baugröße durch Verwendung gemeinsamer Komponenten und mehrerer Prozessorkerne in einem Gehäuse



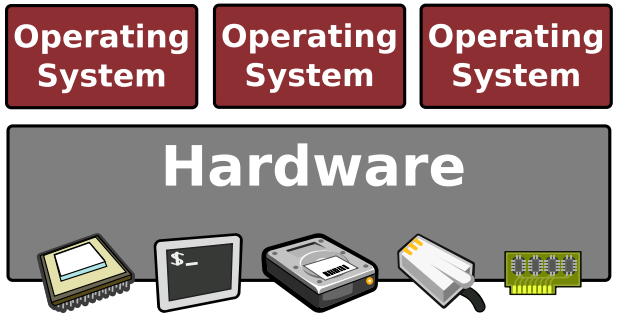
Virtual machine monitor
Type 1

Beispielanwendung

- Ausführung mehrerer Betriebssysteme auf einer Multi-Core-Plattform
- Mehraufwand bei Emulation oder Virtualisierung

Parallel koexistente Betriebssysteme

- Weitestgehend autarke Laufzeitumgebungen auf einer Multi-Core-Plattform
- Sequenzialisierung auf Grund zeitgleicher Zugriffe auf Hardware-Module



parallel coexistent Operating System

Forschungsschwerpunkte

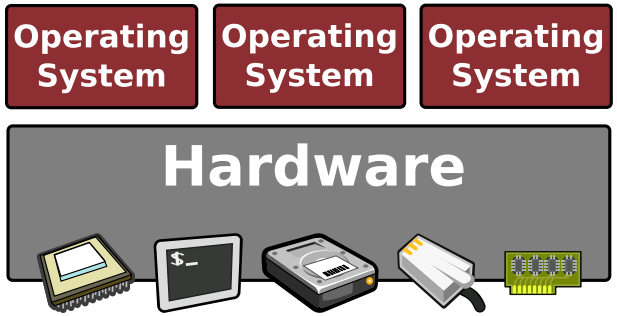
- Peripheriegeräte
- Interrupt-System
- Speicher und Cache
- Betriebssysteminterne Strukturen
- Betriebssystemübergreifende Kommunikation

Arbeitsumgebung

- Modifizierter Linux–Kernel 2.6.22.10
- Entwicklungsplattform mit ARM11 MPCore
- Skripte zur Kompilierung von Toolchain und Kernel

Parallel koexistente Betriebssysteme

- Weitestgehend autarke Laufzeitumgebungen auf einer Multi-Core-Plattform
- Sequenzialisierung auf Grund zeitgleicher Zugriffe auf Hardware-Module



parallel coexistent Operating System

Forschungsschwerpunkte

- Peripheriegeräte
- Interrupt-System
- Speicher und Cache
- Betriebssysteminterne Strukturen
- Betriebssystemübergreifende Kommunikation

Arbeitsumgebung

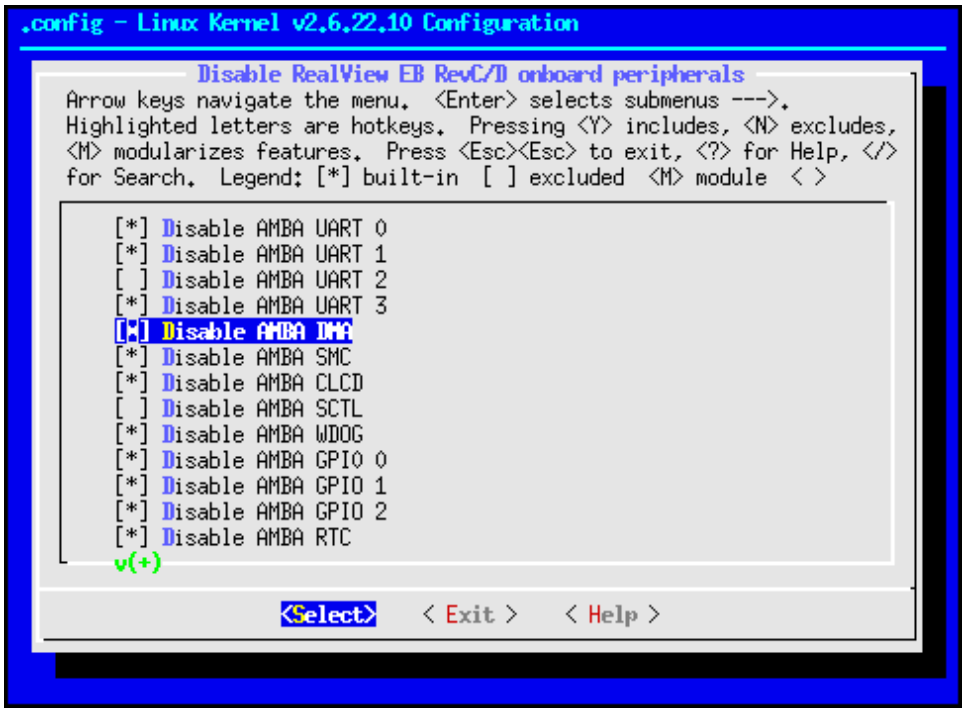
- Modifizierter Linux–Kernel 2.6.22.10
- Entwicklungsplattform mit ARM11 MPCore
- Skripte zur Kompilierung von Toolchain und Kernel

Peripheriegeräte

- Mögliche Wettlaufsituationen durch nebenläufige Zugriffe auf Hardware-Komponenten
- Abhängigkeit der Schutzmechanismen von Zugriffsart

Exklusiver Zugriff

- Kommunikations-Hardware oder Anzeigegeräte wie UART, Netzwerk, DVI, ...
- Ausgewählte Zuordnung zu einem Betriebssystem
- Erweiterung zu 'virtuellen Geräten' durch betriebssystemsübergreifende Kommunikation

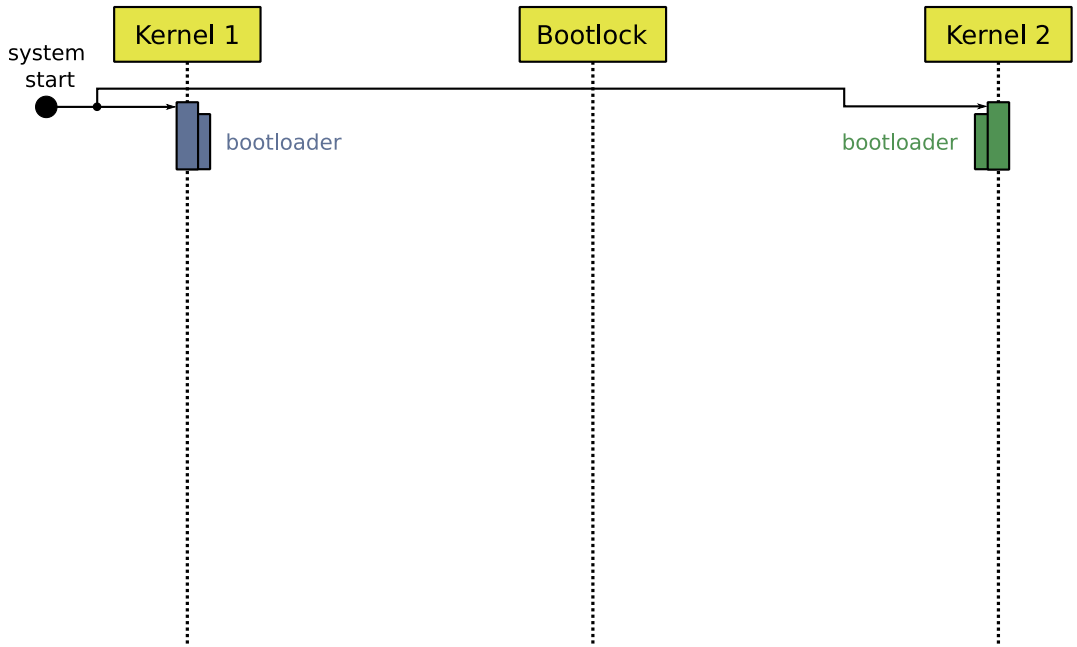


Peripheriegeräte

- Mögliche Wettlaufsituationen durch nebenläufige Zugriffe auf Hardware-Komponenten
- Abhängigkeit der Schutzmechanismen von Zugriffsart

Geteilter Zugriff

- Komponenten zur Systemkonfiguration z.B. Interrupt-Controller, Cache-Controller, ...
- Sperrmechanismen zum wechselseitigen Ausschluss
- Einteilung in einen primären und weitere sekundäre Bootvorgänge

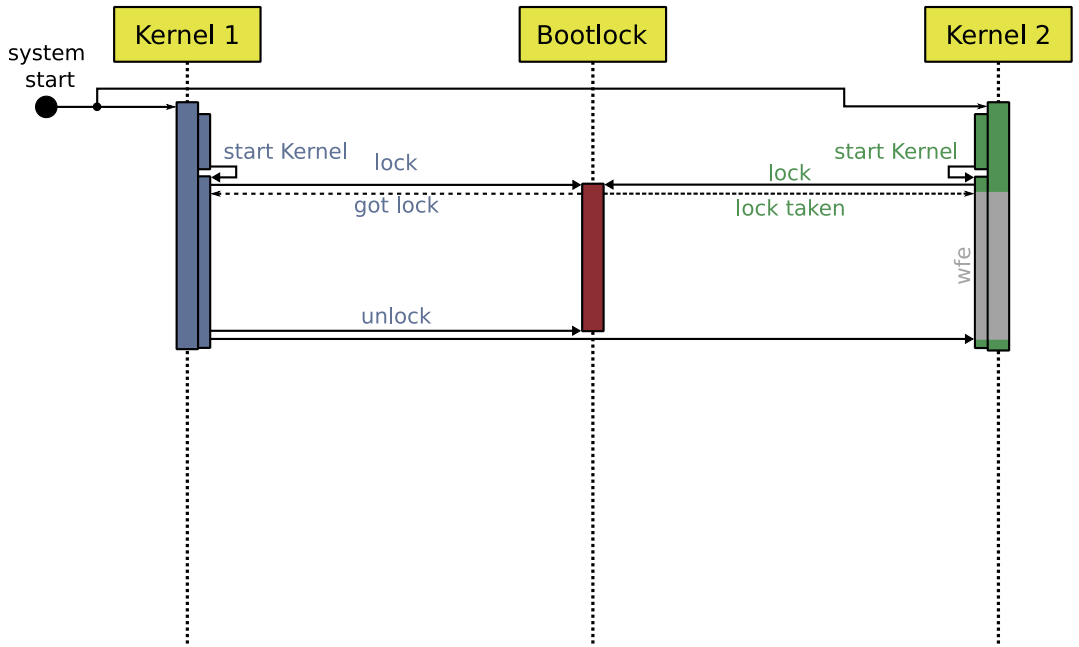


Peripheriegeräte

- Mögliche Wettlaufsituationen durch nebenläufige Zugriffe auf Hardware-Komponenten
- Abhängigkeit der Schutzmechanismen von Zugriffsart

Geteilter Zugriff

- Komponenten zur Systemkonfiguration z.B. Interrupt-Controller, Cache-Controller, ...
- Sperrmechanismen zum wechselseitigen Ausschluss
- Einteilung in einen primären und weitere sekundäre Bootvorgänge

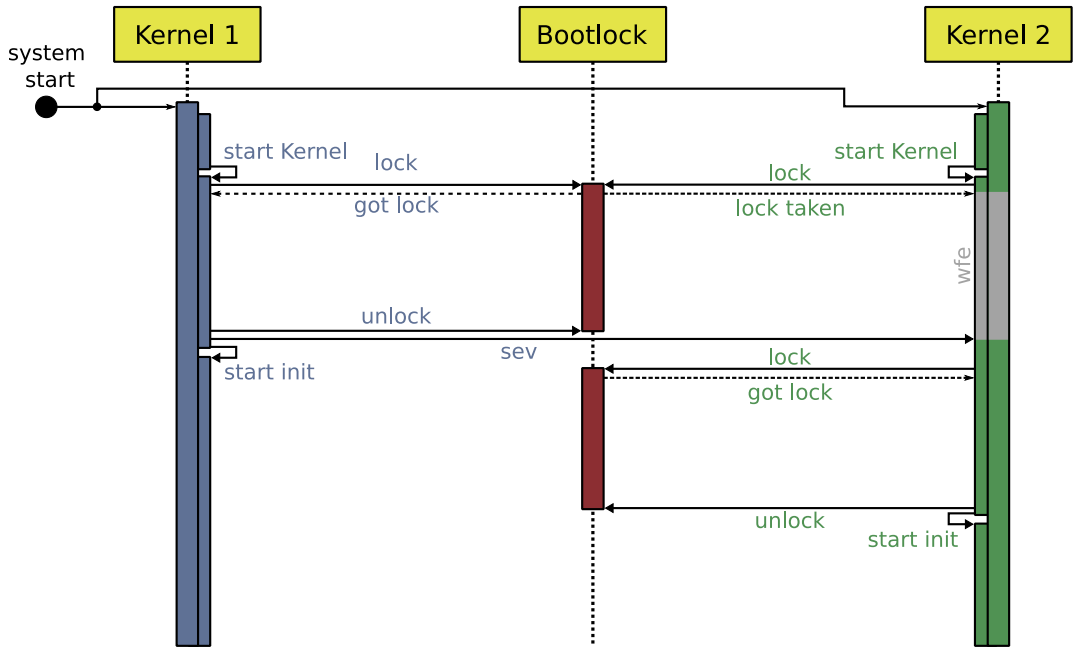


Peripheriegeräte

- Mögliche Wettlaufsituationen durch nebenläufige Zugriffe auf Hardware-Komponenten
- Abhängigkeit der Schutzmechanismen von Zugriffsart

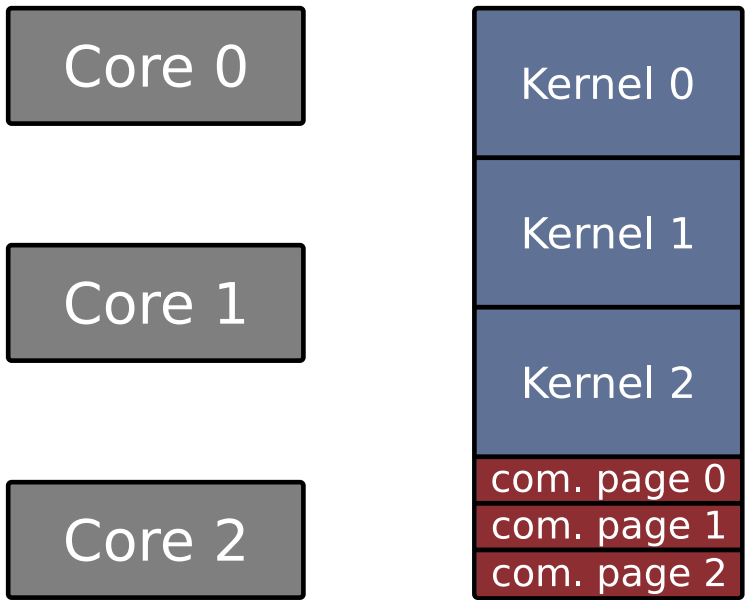
Geteilter Zugriff

- Komponenten zur Systemkonfiguration z.B. Interrupt-Controller, Cache-Controller, ...
- Sperrmechanismen zum wechselseitigen Ausschluss
- Einteilung in einen primären und weitere sekundäre Bootvorgänge



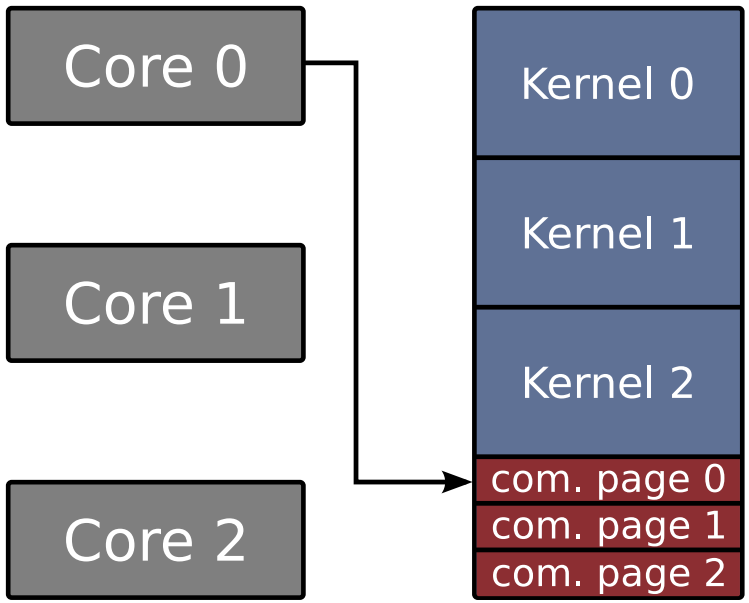
Betriebssystemübergreifende Kommunikation

- Kommunikation über Peripheriegeräte oder spezielle Bereiche im Speicher möglich
- Abhängigkeiten durch stärkere Verflechtung der Betriebssysteme



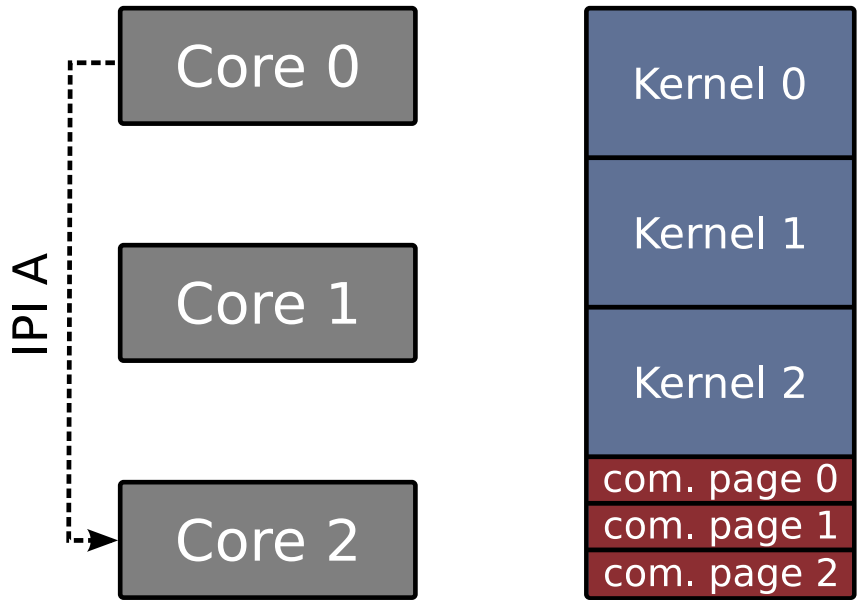
Betriebssystemübergreifende Kommunikation

- Kommunikation über Peripheriegeräte oder spezielle Bereiche im Speicher möglich
- Abhängigkeiten durch stärkere Verflechtung der Betriebssysteme



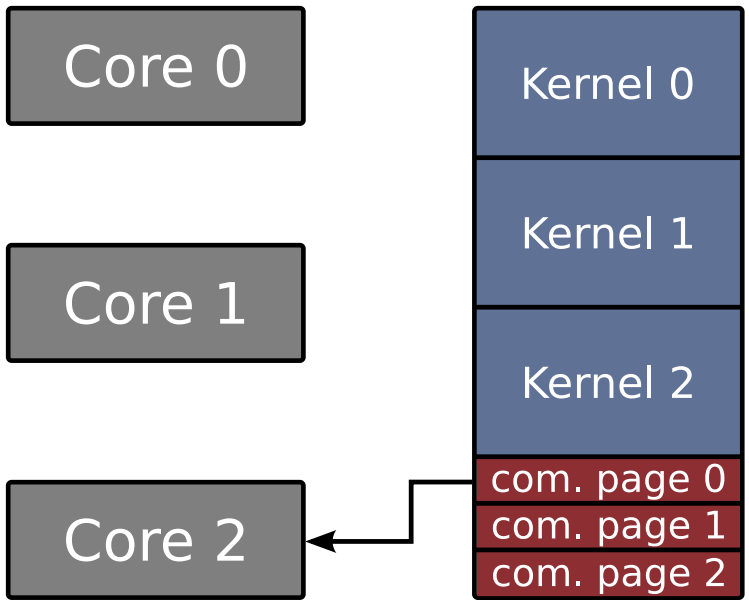
Betriebssystemübergreifende Kommunikation

- Kommunikation über Peripheriegeräte oder spezielle Bereiche im Speicher möglich
- Abhängigkeiten durch stärkere Verflechtung der Betriebssysteme



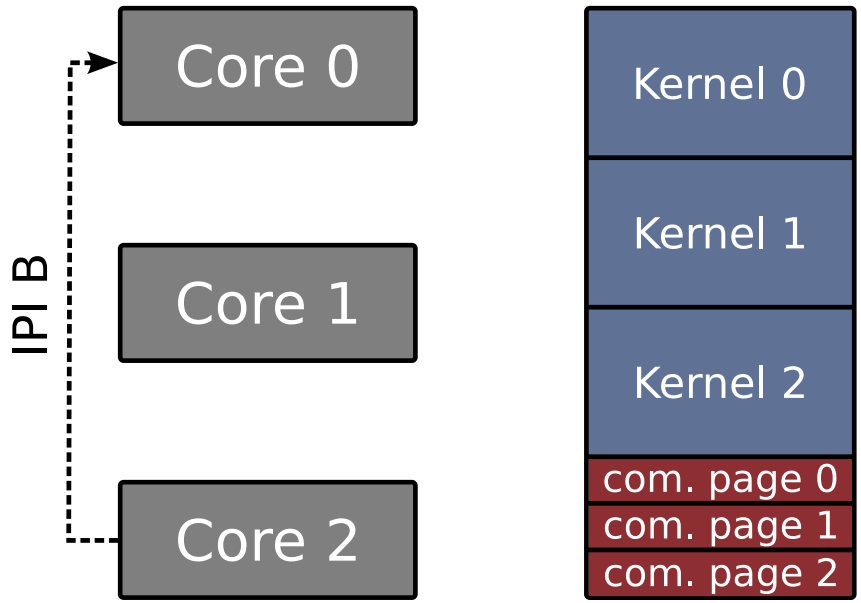
Betriebssystemübergreifende Kommunikation

- Kommunikation über Peripheriegeräte oder spezielle Bereiche im Speicher möglich
- Abhängigkeiten durch stärkere Verflechtung der Betriebssysteme



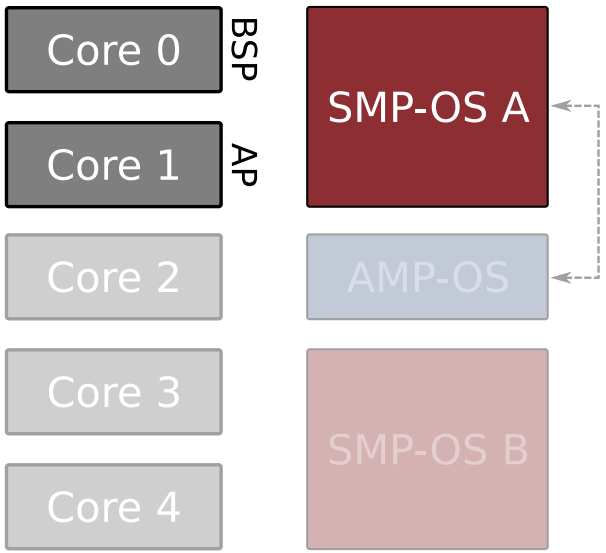
Betriebssystemübergreifende Kommunikation

- Kommunikation über Peripheriegeräte oder spezielle Bereiche im Speicher möglich
- Abhängigkeiten durch stärkere Verflechtung der Betriebssysteme



Debugger-Erweiterungen

- Auswahl und Steuerung der Prozessorkerne ohne System–Reset
- Verbesserung der Fehlersuche durch neue Systemsichten



Arbeitsumgebung

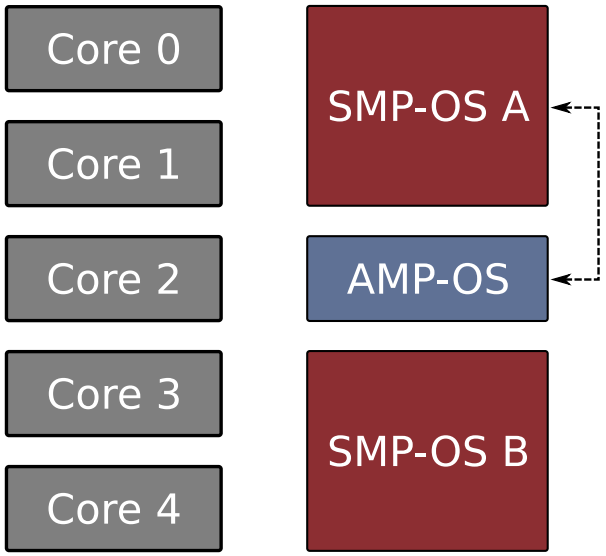
- Universal Debug Engine der Firma pls
- Anbindung über Jtag–Protokoll

Prozessorhierarchie in SMP-Systemen

- 'Boot Strap Processor' initialisiert ein SMP-System
- Starten der 'Application Processors' über plattformspezifischen Mechanismus

Debugger-Erweiterungen

- Auswahl und Steuerung der Prozessorkerne ohne System–Reset
- Verbesserung der Fehlersuche durch neue Systemsichten



Arbeitsumgebung

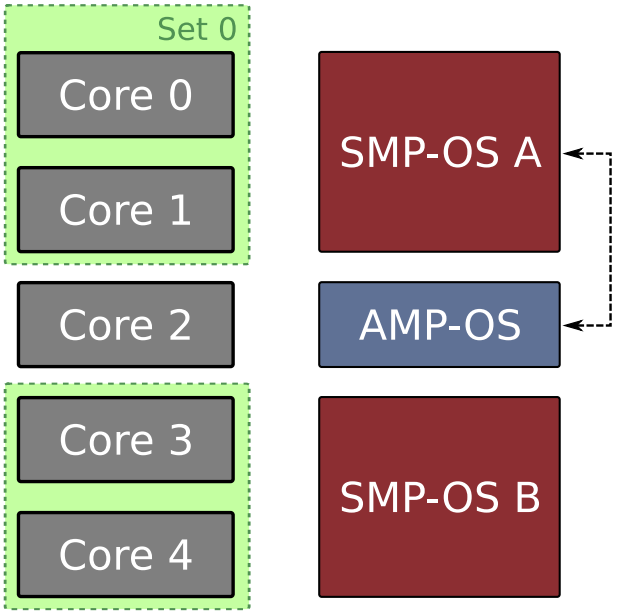
- Universal Debug Engine der Firma pls
- Anbindung über Jtag–Protokoll

Gruppierung von Prozessorkernen

- Zusammenfassung einzelner Prozessorkerne zu 'Sets'
- Synchrone Ausführung von Debug-Kommandos auf alle Prozessorkerne in einem 'Set'

Debugger-Erweiterungen

- Auswahl und Steuerung der Prozessorkerne ohne System–Reset
- Verbesserung der Fehlersuche durch neue Systemsichten



Arbeitsumgebung

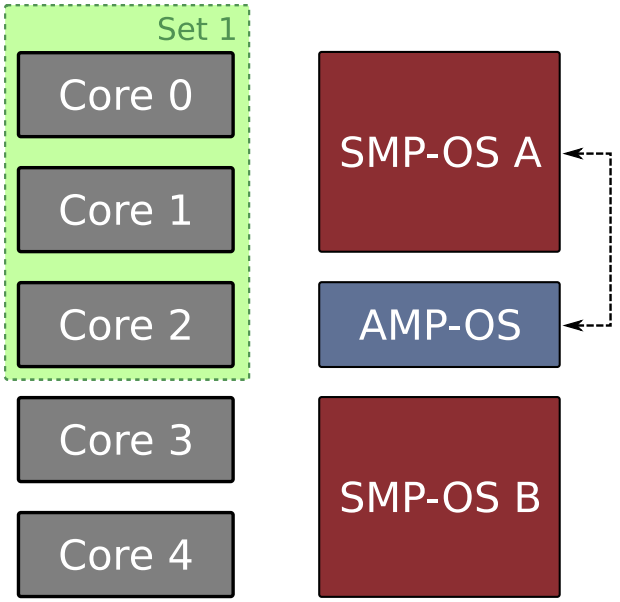
- Universal Debug Engine der Firma pls
- Anbindung über Jtag–Protokoll

Gruppierung von Prozessorkernen

- Zusammenfassung einzelner Prozessorkerne zu 'Sets'
- Synchrone Ausführung von Debug-Kommandos auf alle Prozessorkerne in einem 'Set'

Debugger-Erweiterungen

- Auswahl und Steuerung der Prozessorkerne ohne System–Reset
- Verbesserung der Fehlersuche durch neue Systemsichten



Arbeitsumgebung

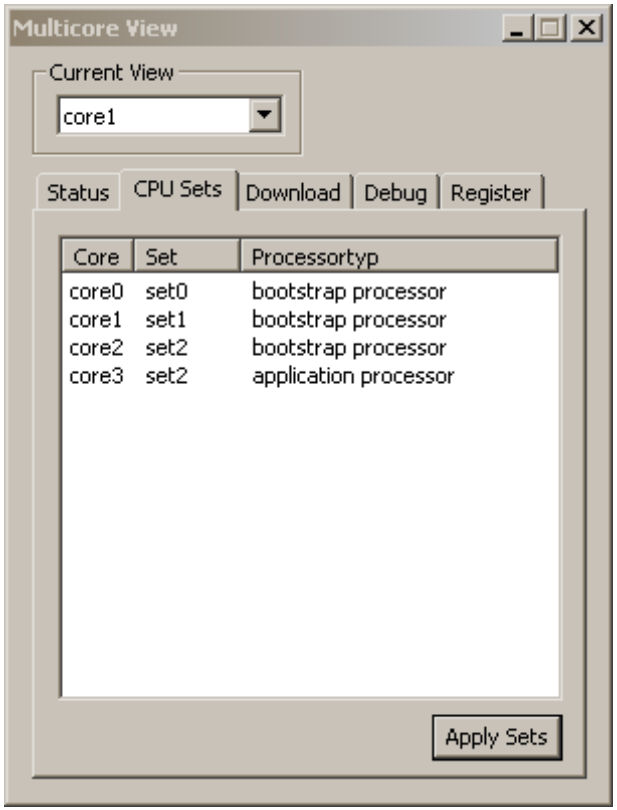
- Universal Debug Engine der Firma pls
- Anbindung über Jtag–Protokoll

Gruppierung von Prozessorkernen

- Zusammenfassung einzelner Prozessorkerne zu 'Sets'
- Synchrone Ausführung von Debug-Kommandos auf alle Prozessorkerne in einem 'Set'

Debugger-Erweiterungen

- Auswahl und Steuerung der Prozessorkerne ohne System–Reset
- Verbesserung der Fehlersuche durch neue Systemsichten



Arbeitsumgebung

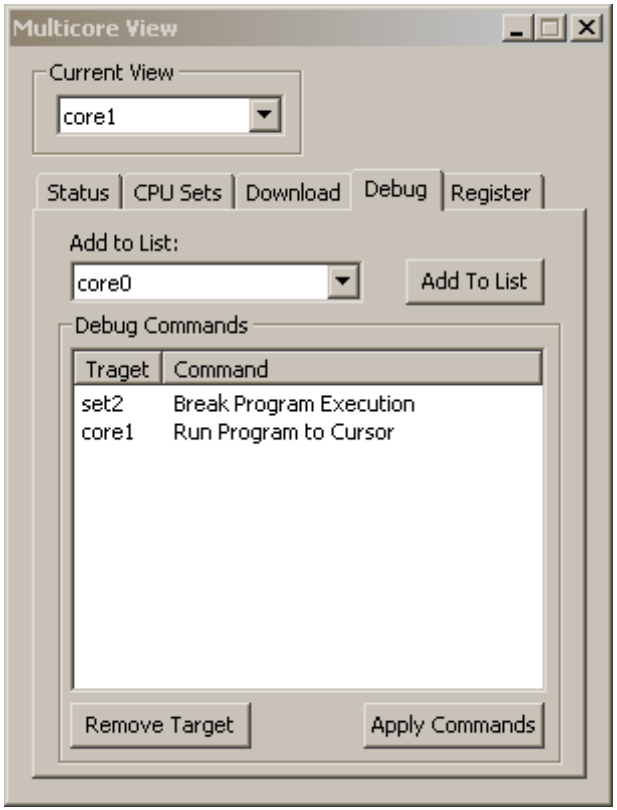
- Universal Debug Engine der Firma pls
- Anbindung über Jtag–Protokoll

Gruppierung von Prozessorkernen

- Zusammenfassung einzelner Prozessorkerne zu 'Sets'
- Synchrone Ausführung von Debug-Kommandos auf alle Prozessorkerne in einem 'Set'

Debugger-Erweiterungen

- Auswahl und Steuerung der Prozessorkerne ohne System–Reset
- Verbesserung der Fehlersuche durch neue Systemsichten



Arbeitsumgebung

- Universal Debug Engine der Firma pls
- Anbindung über Jtag–Protokoll

Gruppierung von Prozessorkernen

- Zusammenfassung einzelner Prozessorkerne zu 'Sets'
- Synchrone Ausführung von Debug-Kommandos auf alle Prozessorkerne in einem 'Set'

Zusammenfassung

- Konzeption und Implementierungen eines Sperrmechanismus, der Boot-Hierarchie, der Zuweisung von Peripheriegeräten und der Interrupt-Weiterleitung
- Noch unvollständiger Start zweier Linux–Kernel
- Entwurf des Gruppierungs- und Hierarchiekonzepts einzelner Prozessorkerne für die Fehlersuche in Multi-Core-Systemen
- Entwicklung eines Kontrollfensters für die UDE mit zusätzlichen Übersichtsfunktionen
- Vorschläge zur betriebssystemsübergreifenden Kommunikation und zur Steuerung von Breakpoints
- Anpassung der maschinennahen Fehlerausgabefunktionen des Linux-Kernels
- Auswahl eines Benchmarks und Durchführung erster Messungen
- Emulation der ARM11 MPCore–Plattform mit Hilfe von Qemu und hardwarenaher Fehlersuche

Ausblick

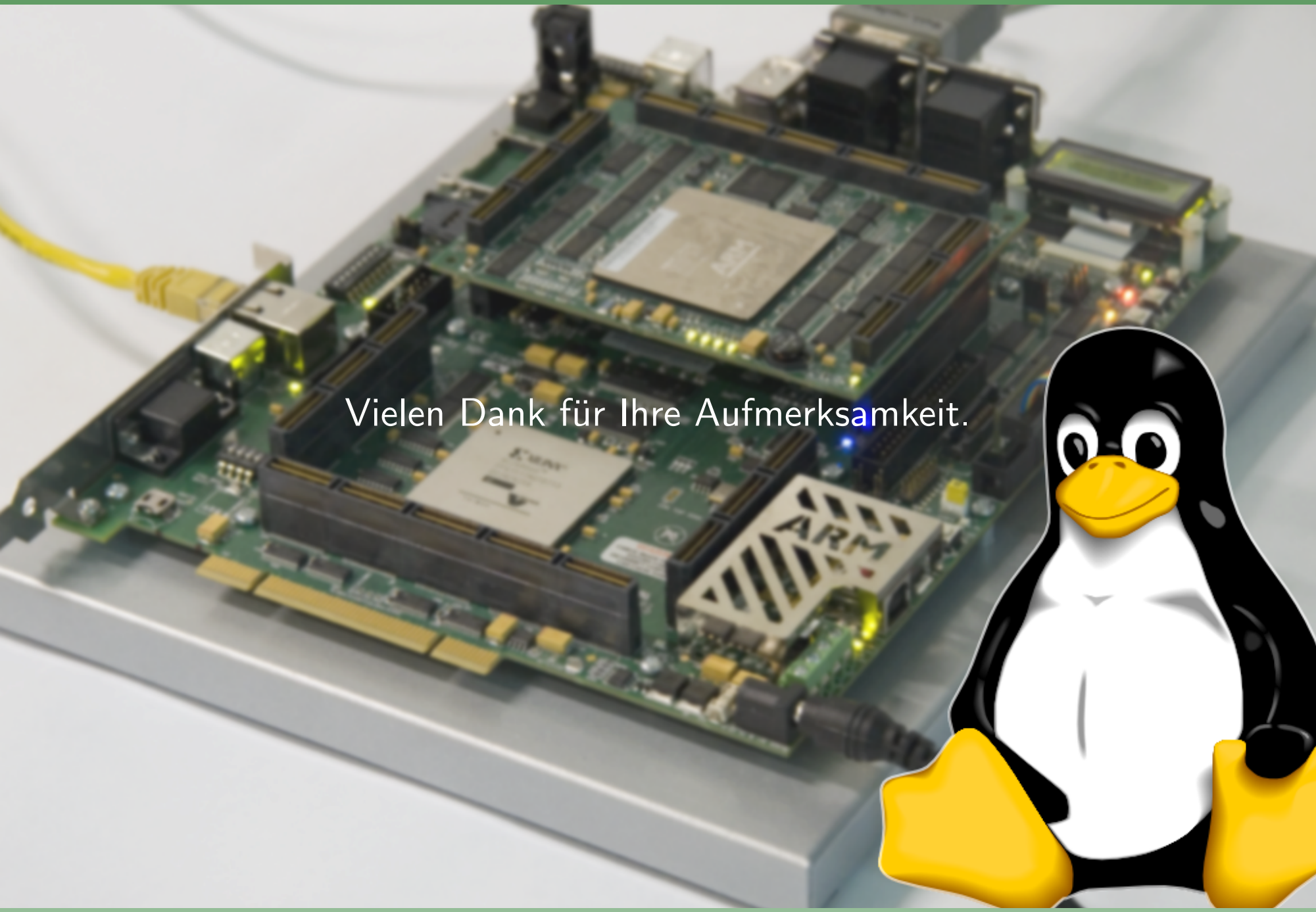
Weiterführende Arbeiten am Linux-Kernel

- Anpassung des Speicherallokationsmechanismus
- Betriebssystemübergreifende Kommunikation
- Verfeinerung des Bootlock–Mechanismus

Weiterführende Arbeiten am Debugger

- Vollständige Integration des Kontrollfensters in die UDE
- Anpassung der Breakpoint-Funktionen an das Prinzip der Sets
- Integration von Fehlersuchmechanismen des Betriebssystems

Nach Abschluss der Implementierung dieses Prototyps ist es erforderlich, die vorgestellten Mechanismen auf andere Betriebssysteme anzuwenden, sowie eine Leistungsbewertung mittels Benchmarks durchzuführen.



Vielen Dank für Ihre Aufmerksamkeit.