

# Echtzeitsysteme

## Mehrkern-Echtzeitsysteme

**Peter Ulbrich**

Lehrstuhl für Verteilte Systeme und Betriebssysteme  
Friedrich-Alexander-Universität Erlangen-Nürnberg

<https://www4.cs.fau.de>

27. Januar 2016





Wie lassen sich **komplexe Echtzeitsysteme** handhaben?

- **Rechenzeitbedarf** ist durch einfache Rechensysteme nicht zu erfüllen
- Beispiel: Moderne Fahrerassistenzsystemen
- Die **Vielfalt** der abzuarbeitenden Aufgaben ist enorm

■ **Herausforderungen** von Mehrkern-Echtzeitsystemen?

- Welche Anomalien entstehen durch Ausführungsparallelität?
- Welche Konsequenzen hat dies für die Ablaufplanung?

 Wie sehen **Verfahren** und **Mechanismen** für Mehrkern-Echtzeitsystem aus?

- Ablaufplanung?
- WCET-Analyse?
- Zugriffssteuerung?



- Liu-Layland Planbarkeitskriterium für statische Prioritäten<sup>1</sup>
  - RMA ist optimal für periodische Aufgabensysteme (vgl. IV-2/5 ff)
  - Planbare Auslastung  $u_{RMA} \leq \ln(2) \sim 69,3\%$
- Planungsalgorithmen für dynamische Prioritäten sind optimal<sup>1</sup>
  - Beispielsweise EDF für beliebige Aufgabensysteme (vgl. IV-2/13 ff)
- Ablaufplanung behält auch im positiven Fall ihre Zulässigkeit
  - Wenn sich das System besser verhält als angenommen
  - Antwortzeiten vergrößern sich nicht bei abnehmender Ausführungszeit
  - Auslösezeiten und Termine verschieben sich ausführungsbedingt nicht
- Gleichzeitige Auslösung repräsentiert den kritischen Zeitpunkt
  - Maximale Antwortzeit hängt von der Menge der Aufgaben ab (vgl. IV-2/36)



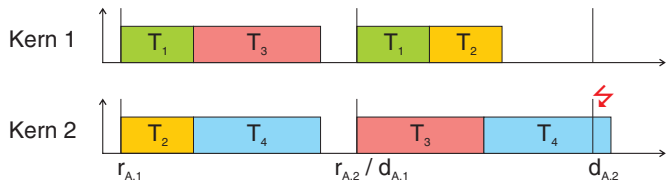
Viele in Einkernsystemen valide Annahmen verlieren in Mehrkernsystemen ihre Gültigkeit!

<sup>1</sup>Bezogen auf die Priorität der Aufgaben (task priority).



# Anomalie: Kritischer Zeitpunkt

Antwortzeit in Abhängigkeit von der Ausführungsreihenfolge



Gleichzeitige Auslösung repräsentiert nicht mehr zwingend den **kritischen Zeitpunkt**

- Antwortzeit der zweiten Periode vergrößert
- Terminverletzung durch Wahl des Kerns



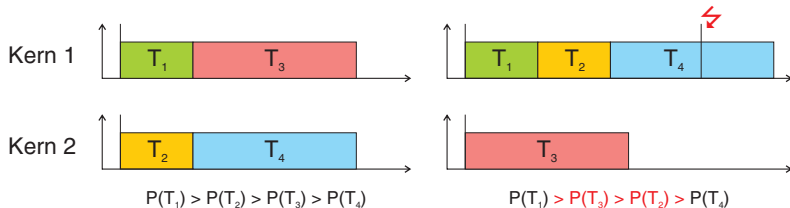
**Dhall Effekt [3]**

- Garantierte Auslastung wird beliebig schlecht
- Konvergiert im schlimmsten Fall gegen  $u = 1$  (Einkernsystem)



# Anomalie: Relative Prioritätsordnung

Antwortzeit in Abhängigkeit von der Ausführungsreihenfolge



⚠ Antwortzeit abhängig vom relativen Prioritätsgefüge

- Höherpriorer Arbeitsaufträge
  - Beispiel:  $T_4$  verpasst seinen Termin
    - Vorgezogene Ausführung von  $T_3$  auf Kern 2
    - $T_4$  wird auf Kern 1 eingeplant und verpasst seinen Termin
- Erschwert signifikant die Antwortzeitanalyse



1 Überblick & Motivation

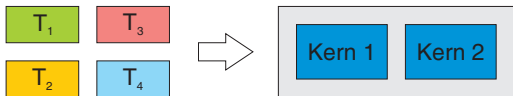
**2** Ablaufplanung

3 WCET-Analyse

4 Zusammenfassung



# Herausforderung Mehrkernechtzeitsystem



## 1 Prioritätsproblem (engl. *priority problem*)

- Wann und in welcher Reihenfolge laufen Arbeitsaufträge?
- Statische Prioritäten für Aufgaben (z.B. RMA)
  - Dynamische Prioritäten für Aufgaben (z.B. EDF)
  - Dynamische Prioritäten für Aufträge (z.B. PFAIR)

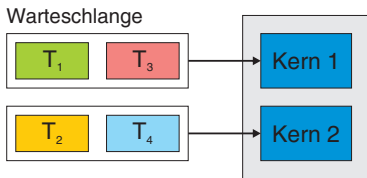
## 2 Allokationsproblem (engl. *allocation problem*)

- Auf welchem Kern laufen Arbeitsaufträge?
- Keine Migration (engl. *no migration*)
  - Migration von Aufgaben (engl. *task-level migration*)
  - Migration von Arbeitsaufträgen (engl. *job-level migration*)



Partitionierte, globale und hybride Ablaufplanungsverfahren





## Partitionierte Ablaufplanung (engl. *partitioned scheduling*)

- **Verteilung der Aufgaben** auf Kerne vor der Laufzeit
  - Alle Aufträge einer Aufgabe werden auf demselben Kern ausgeführt
- **Anwendung klassischer EK-Verfahren** auf die lokale Aufgabenmenge

## ■ Verteilung → **Behälterproblem** (engl. *bin packing problem*)

- Verteile  $n$  Aufgaben der Dichte  $\Delta_i$  auf  $m$  Kerne der Kapazität  $\Delta_{max} = 1$
- Zahlreiche Verfahren verfügbar: **First-Fit**, **Next-Fit**, **Best-Fit**, ...

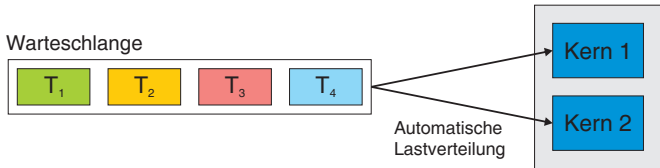


**Planbare Auslastung** im schlimmsten Fall  $u_{wc} \approx 0,5$  (50%)

→ Alle Aufgaben haben eine Auslastung wenig größer als  $u_i > 0,5$







### ☞ Globale Ablaufplanung (engl. *global scheduling*)

- Verteilung der Arbeitsaufträge auf Kerne zur Laufzeit

→ Aufgaben bzw. Aufträge können zwischen Kernen migrieren

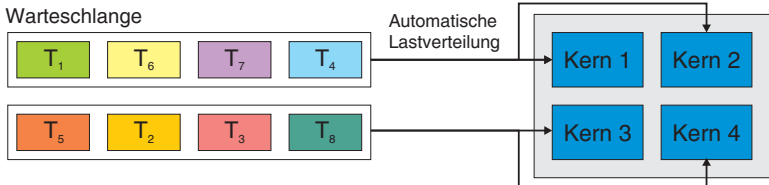
### ■ Mehrkern-Planungsverfahren auf der globalen Warteschlange

- Statische und dynamische Prioritäten: G-EDF, D-RMA, PFAIR, ...
- Verfahren mit dynamic-job-level Prioritäten (z.B. PFAIR) dominieren alle anderen Planungsverfahren  $\leadsto$  Auslastung  $u_{opt} = 1$

⚠ Globale Verfahren können mit erheblichen **Kosten** und **Unwägbarkeiten** behaftet sein

- Hohe Migrationskosten, insbesondere bei Migration von Aufträgen
- Analysierbarkeit der Laufzeit nicht mehr praktikabel





### Hybride Ablaufplanung (engl. *hybrid scheduling*)

- Kombination globaler und partitionierter Ablaufplanung
- Migration minimieren und Auslastung maximieren

### ■ Gruppierende Ablaufplanung (engl. *clustered scheduling*)

- Zusammenfassen von Kernen zu Gruppen (engl. *cluster*) und Partitionierung
- Pro Gruppe eine globale Warteschlange
- Hierarchische Ablaufplanung (vgl. Zusteller V-2/27)

### ■ Teilpartitionierende Ablaufplanung (engl. *semi-partitioned scheduling*)

- Einschränkung der Migration durch (komplexes) Regelwerk
- Beispiele: Aufgaben dürfen nur zwischen bestimmten Kernen migrieren; Begrenze Zahl migrierbarer Aufgaben



1 Überblick & Motivation

2 Ablaufplanung

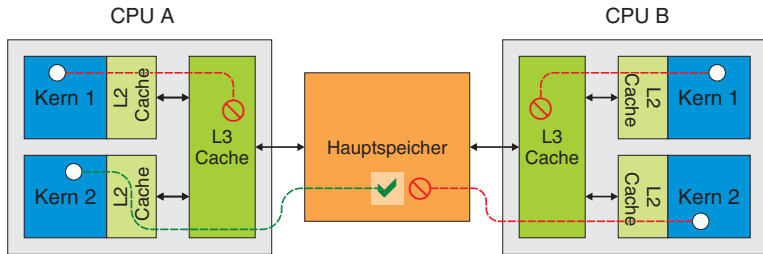
**3 WCET-Analyse**

4 Zusammenfassung



# Herausforderung: Echte Ausführungsparallelität

Interferenz auf der physikalischen Ebene



➔ Echte Ausführungsparallelität ist das Problem

- Einkernsysteme  $\leadsto$  Virtuelle Parallelisierung, Sequentialisierung
- Mehrkernsysteme  $\leadsto$  Gemeinsames Betriebsmittel

■ Beispiel: Simultaner Speicherzugriff

- Impliziter Wettstreit auf physikalischer Ebene

⚠ Auswirkungen auf Laufzeitverhalten

- Blockadezeit durch Speicherzugriff und Invalidation des Cache-Inhalts
- Studien zeigen dramatische Zunahme der WCET [1]

→ Abhängig von den Eigenschaften der Hardware!



### 1 Kosten durch Verdrängung (engl. *preemption costs*)

→ Die Problem sind uns bereits bekannt (vgl. IV-2/?? und III-3), die **Komplexität** der Analyse bzw. Abstraktion **steigt jedoch im Mehrkernfall dramatisch an**

#### ■ Direkte Kosten

- Unterbrechung, Sicherung und -wiederherstellung des Hardwarekontexts
- ⚠ Einplanung  $\leadsto$  **Synchronisation** im Mehrkernsystem notwendig

#### ■ Indirekte Kosten

- Wiederaufsetzen / -füllen der Seitentabelle
- Dislokation von Cacheinhalten (engl. *cache evictions*) zwischen Ausführungsrunden
- ⚠ Ggf. Herstellen der **Cachekoherenz** (engl. *cache coherency*)
- ⚠ Kommunikation zwischen Kernen/Sockeln



### 2 Kosten durch Migration (engl. *migration costs*)

⚠ Diese Kosten entstehen nur in Mehrkernsystemen

#### ■ Direkte Kosten

- Manipulation der **Bereitliste** (engl. *ready queue*)  $\leadsto$  Synchronisation
- Sicherung und -wiederherstellung des Hardwarekontexts

#### ■ Indirekte Kosten

- Laden des aktiven Cachekontextes (engl. *cache working set*) vom Quellkern  $\leadsto$  Dislokation von Cacheinhalten
- Laden des restlichen Prozesskontexts (transitive Hülle)  $\leadsto$  Cachekohärenz, Kommunikation

⚠ Zusätzlich Kosten durch Verdrängung bei Migration von (laufenden) Arbeitsaufträgen



1 Überblick & Motivation

2 Ablaufplanung

3 WCET-Analyse

**4 Zusammenfassung**



## ■ Mehrkernechtzeitsysteme sind die Zukunft

- Leistungssteigerung durch Parallelisierung

## ■ Ablaufplanung ist eine Herausforderung

- Wissen aus Einkernsystemen im Allgemeinen nicht übertragbar
- Zeitliche Anomalien  $\rightsquigarrow$  Kritischer Zeitpunkt, Prioritätsordnung
- Prioritätsproblem und Allokationsproblem

## ■ Partitionierte Ablaufplanung

- Verteilen der Aufgaben auf Kerne zum Entwurfszeitpunkt
- Transformation in mehrere Einkernsysteme
- Bekannte Techniken und Algorithmen sind wieder anwendbar
- Garantiert planbare Auslastung sehr schlecht

## ■ Globale Ablaufplanung

- Findet zur Laufzeit statt und erfordert Migration
- Verfahren mit dynamischen Prioritäten auf Auftragsebene erlauben vollständige Auslastung
- In der Praxis mit hohen Kosten und Unwägbarkeiten behaftet





- **Hybride Ablaufplanung**
  - Verbinden die Vor- und Nachteile der anderen Verfahren
  - Teilpartitionierte und Gruppierende Ablaufplanung
- **WCET-Analyse**
  - Komplexität nimmt stark zu
  - Zusätzliche Kosten durch Migration und Synchronisation



- [1] Baruah, S. ; Bertogna, M. ; Buttazzo, G. :  
*Multiprocessor Scheduling for Real-Time Systems*.  
Springer, 2015. –  
ISBN 978–3319086958
  
- [2] Davis, R. I. ; Burns, A. :  
A Survey of Hard Real-Time Scheduling for Multiprocessor Systems.  
In: *ACM Computing Surveys* 43 (2011), Okt., Nr. 4.  
<http://dx.doi.org/10.1145/1978802.1978814>. –  
DOI 10.1145/1978802.1978814
  
- [3] Dhall, S. K. ; Liu, C. :  
On a real-time scheduling problem.  
In: *Operations research* 26 (1978), Nr. 1, S. 127–140

