TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Lehrstuhl Betriebssysteme

Effiziente Unterstützung von Multiprozessorsystemen im Fiasco-Mikrokern unter Beachtung des zeitlichen Ausführungsverhaltens

Matthias Lange

Dresden, 21. September 2007

Trend zu Multiprozessoren

- Multicore- und Multiprozessorsysteme Trend in der Halbleiterindustrie
  - weitere Entwicklungen in dieser Richtung erwartet
- Ausnutzung von Task-Parallelität um Geschwindigkeit zu steigern
- Echtzeitaufgaben können von MP profitieren
  - explizites MP-Modell
  - Partitionierung

TECHNISCHE UNIVERSITÄT DRESDEN

Überblick

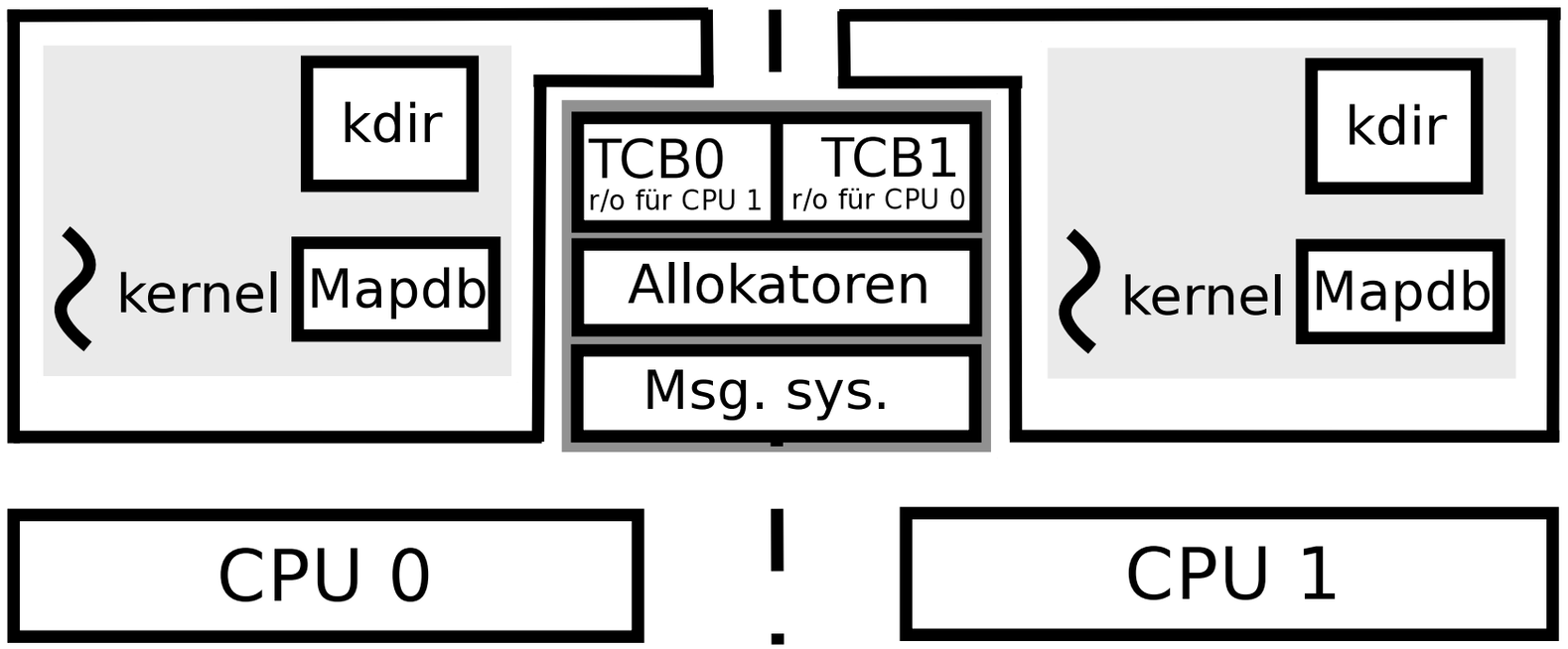
- Motivation
- FiascoMP 2.1
- Design
- Implementierung
- Auswertung
- Zusammenfassung

FiascoMP 2.1

- Unterstützung von Multiprozessorsystemen in Fiasco
  - potentielle Unterstützung für L4LinuxMP
  - Erhalt der Echtzeiteigenschaften der Uniprozessorvariante
  - Limitierungen der aktuellen Variante:
    - Tasks auf einen Prozessor beschränkt
    - keine Speicher-Mappings über Prozessorgrenzen hinweg
      - Duplikation von Infrastruktur
  - wenige Änderungen gegenüber der Uniprozessorvariante

TECHNISCHE UNIVERSITÄT DRESDEN

FiascoMP 2.1 Architektur



TECHNISCHE UNIVERSITÄT DRESDEN

Weiterentwicklung

- Vermeidung von Duplikation auf Kern- und Benutzerebene
  - Ressourcenverbrauch
  - Laufzeitmehraufwand
- Leistungserhaltung lokaler Operationen
  - Anwendungen sind partitioniert
- Zeiteigenschaften
  - Echtzeiteigenschaften
  - kein Verhungern
- ausreichende Funktionalität für L4LinuxMP

TECHNISCHE UNIVERSITÄT DRESDEN

Überblick

- Motivation
- FiascoMP 2.1
- Design
- Implementierung
- Auswertung
- Zusammenfassung

Synchronisation

- drei verschiedene Synchronisationsfälle:

IPC

- Geschwindigkeitskritisch

Thread-Manipulation

- Deadlock-Problematik

globale Ressourcen

- potentiell lange Zugriffszeiten, kritisch für Echtzeitverhalten
- Fairness beim Zugriff

Synchronisationsansätze

- atomare Ausführung
- Synchronisation über Locks
  - Spin-Lock
  - Helping-Lock
    - Unterbrechbarkeit
    - Fortschritt von lauffähigen Threads

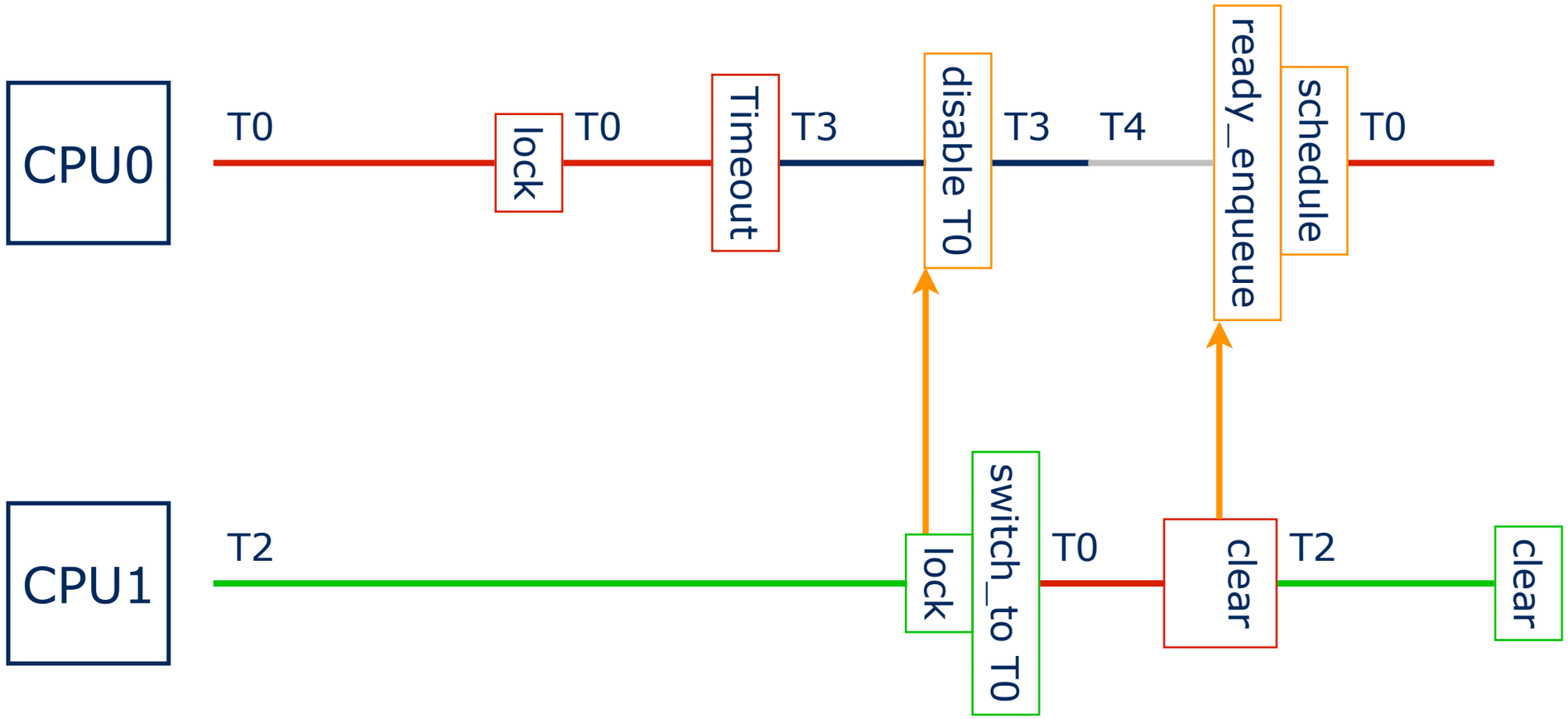
Objektsynchronisation

- Thread-Manipulation und IPC
  - lokale atomare Ausführung
  - Message-Box-System zur Initiierung entfernter atomarer Ausführung
- globale Ressourcen
  - Helping-Lock
    - Unterstützung für Multiprozessoren ist notwendig

Cross-Prozessor-Tasks

- Threads einer Task können auf unterschiedlichen Prozessoren laufen
- TLB-Konsistenzprotokol benötigt
- Remote-Ex-Regs
  - Thread-Lock-Semantik im MP-Fall unterschiedlich zum Uniprozessorfall
    - Deadlocks bei gegenseitigem Ex-Regs über Prozessorgrenzen möglich

Beispiel


CPU0
T0
lock
T0
Timeout
T3
disable T0
T3
T4
ready_enqueue
schedule
T0
CPU1
T2
lock
switch_to T0
T0
clear
T2
clear

TECHNISCHE UNIVERSITÄT DRESDEN

Überblick

- Motivation
- FiascoMP 2.1
- Design
- Implementierung
- Auswertung
- Zusammenfassung

Cross-Prozessor-Tasks

- globale Kern-Seitentabelle
  - prozessorlokale Daten werden mit Hilfe eines C++-Templates definiert
    - beim Start wird Speicher für prozessorlokale Daten reserviert
      - Timeout-Listen, FPU-Besitzer, TSS
- globale Ressourcen
  - Mapping-Datenbank
  - Allokatoren
- Ex-Regs-Systemruf
  - über ein Flag wird eine Remote-Operation signalisiert
  - Erweiterung des Message-Box-Systems um einen neuen Nachrichtentyp für Remote-Ex-Regs

MP-Preemption-Lock

- Erweiterung von bestehenden Subsystemen notwendig
  - `Context`: Verwaltung des Zustands
  - `Cpu`: Unterbrechung und Benachrichtigung von Helfern
  - Unterbrechungspfade (IRQ, Timer) modifiziert
- Assistenten-Thread
  - übernimmt Aufgaben die nicht im aktuellen Thread-Kontext beendet werden können

TECHNISCHE UNIVERSITÄT DRESDEN

Überblick

- Motivation
- FiascoMP 2.1
- Design
- Implementierung
- Auswertung
- Zusammenfassung

Geschwindigkeitsgewinn

- Geschwindigkeitsgewinn durch parallele Nutzung mehrerer Prozessoren
- ideal: linearer Anstieg mit der Anzahl der Prozessoren
- Messprogramm:
  - multithreaded Berechnung der Mandelbrotmenge
- Mess-Hardware
  - Pentium 3, zwei Prozessoren, 450MHz, 256MB RAM
  - Pentium D, Dual-Core, 3GHz, 512MB RAM
  - Pentium D, 2x2 mit Hyperthreading, 2,66GHz, 256MB RAM

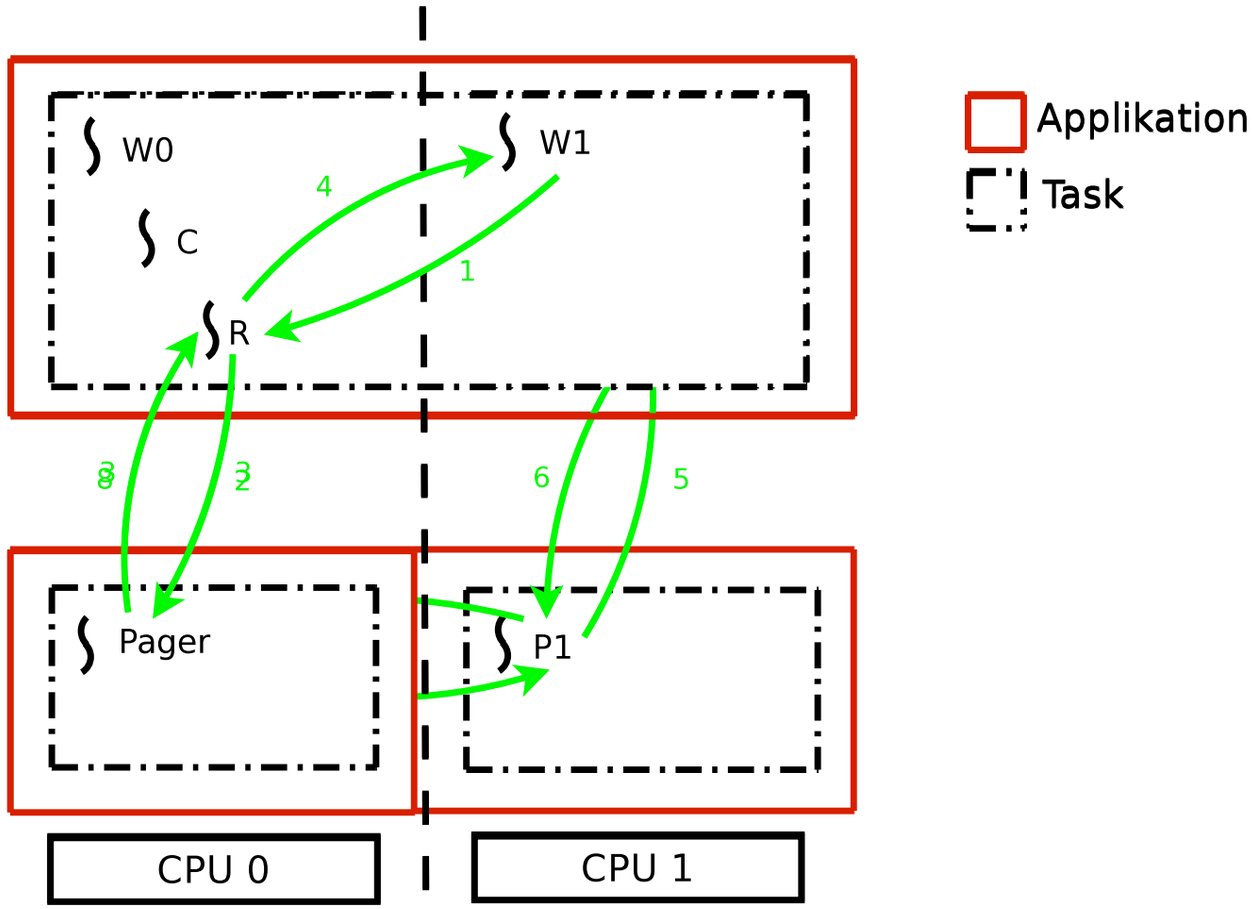
Geschwindigkeitsgewinn

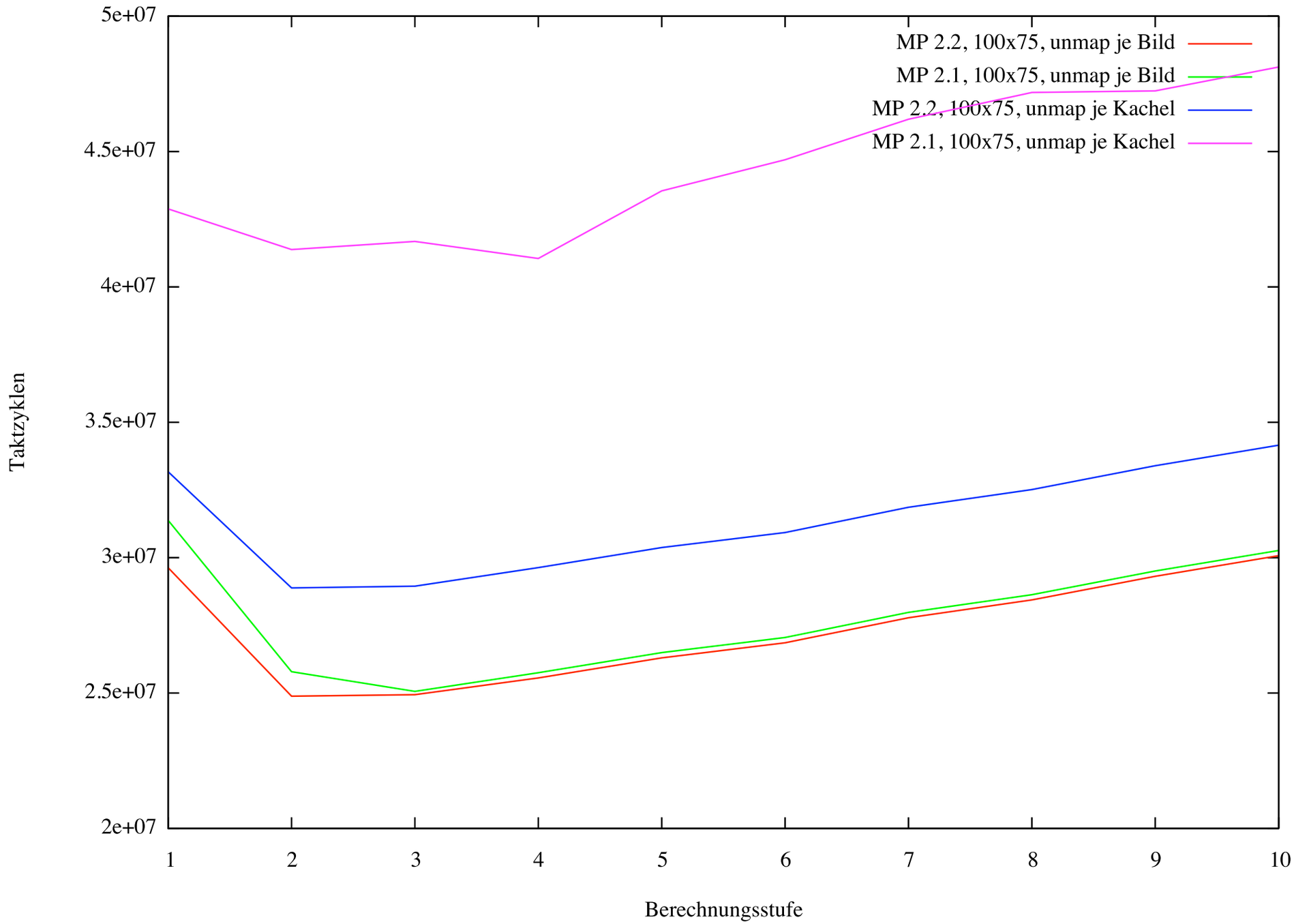
- gemessen auf Pentium 3 und Pentium D 2x2 mit HT

Prozessoren	Zuwachs
2	2
4 (mit HT)	3,83
4 (ohne HT)	3,90
8	7,66

Dynamische Arbeitslast

- dynamische Zuteilung von Speicher
- Verhältnis von Berechnungsdauer und Häufigkeit von Operationen der Speicherverwaltung ändern
- Vergleich der alten und der neuen Implementierung




MP 2.2, 100x75, unmap je Bild
MP 2.1, 100x75, unmap je Bild
MP 2.2, 100x75, unmap je Kachel
MP 2.1, 100x75, unmap je Kachel
Taktzyklen
5e+07
4.5e+07
4e+07
3.5e+07
3e+07
2.5e+07
2e+07
1
2
3
4
5
6
7
8
9
10
Berechnungsstufe

IPC-Geschwindigkeit

- Pingpong-Benchmark gemessen auf dem Pentium-3-System

Testname	Uniprozessor-Fiasco	FiascoMP
int30/warm	1344	1393
sysenter/warm	1165	1165

Unabhängigkeit lokaler Operationen

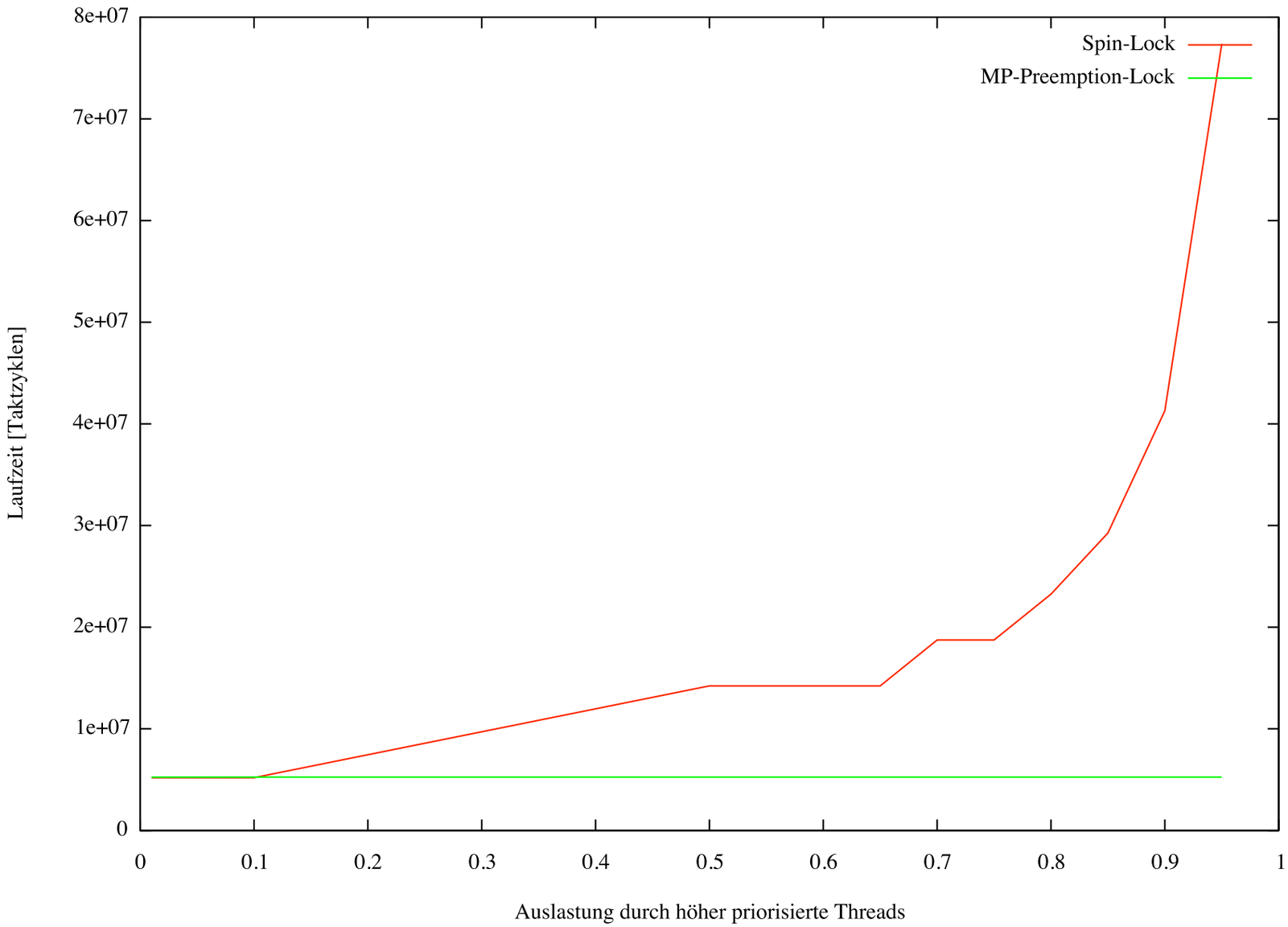
- selbst geschriebenes IPC-Pingpong, lokal auf jedem Prozessor
- gemessen auf dem Pentium-D-2x2-System

Prozessor #	Takte
1	2370
2	2103
3	2528
4	2030
5	2402
6	2001
7	2209
8	2047

MP-Preemption-Lock

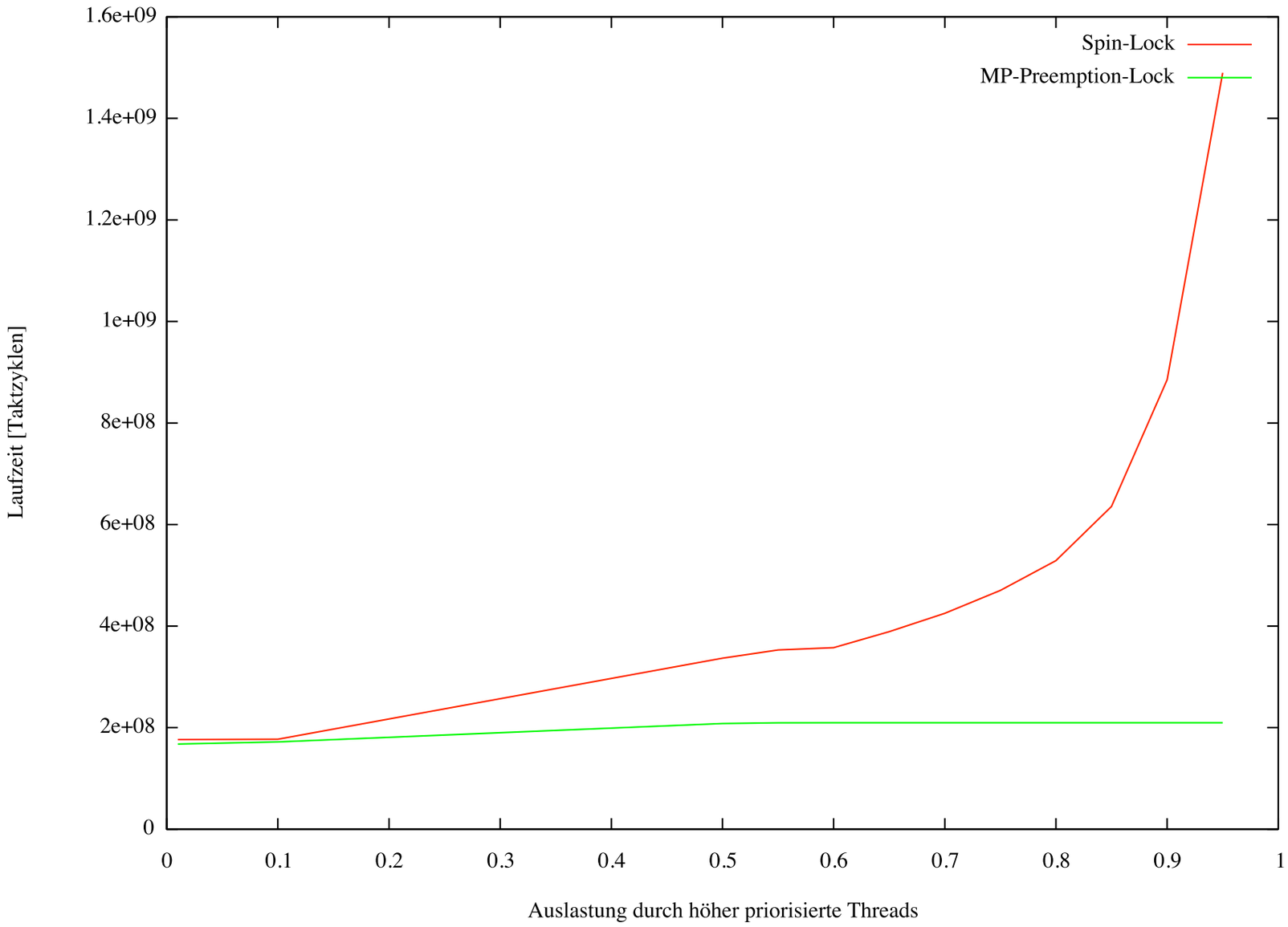
- Vergleich mit unterbrechbarem Spin-Lock
- Messung der maximalen Verzögerungszeit
- Messung der maximalen und durchschnittlichen Gesamtausführungszeit des kritischen Abschnitts
- Messszenario:
  - auf Prozessor 0 laufen 2 Threads, Thread 0 läuft mit hoher Priorität und greift kein Lock, Thread 1 läuft mit niedriger Priorität und konkurriert um ein Lock
  - auf Prozessor 1 läuft 1 Thread der um das Lock konkurriert
  - Thread 1 auf Prozessor 0 wird durch Thread 0 verdrängt, die Auslastung durch Thread 0 wird schrittweise erhöht

Verzögerung (max.)

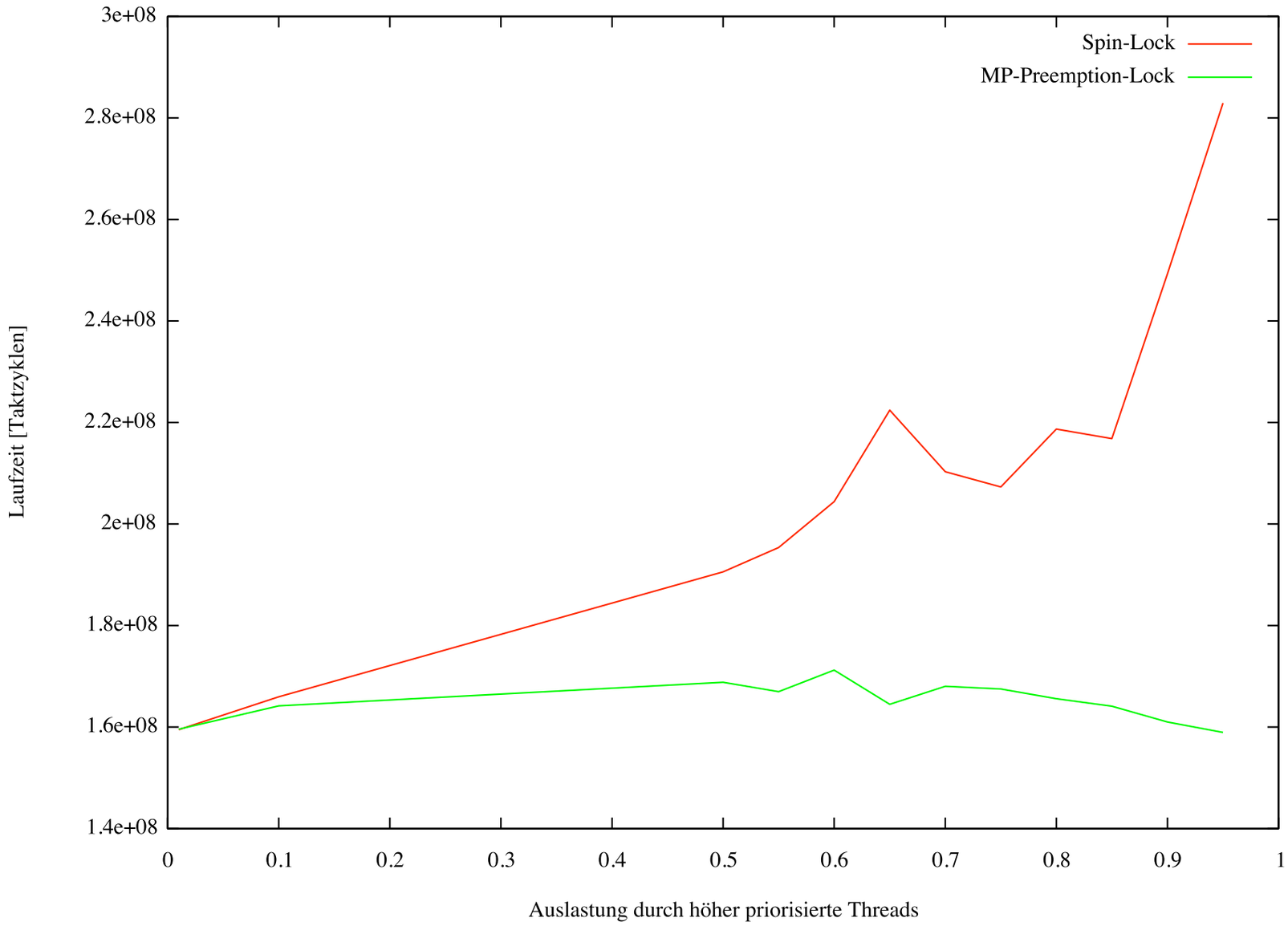


TECHNISCHE UNIVERSITÄT DRESDEN

Gesamtzeit (max.)



Gesamtzeit (durchschn.)



TECHNISCHE UNIVERSITÄT DRESDEN

Überblick

- Motivation
- FiascoMP 2.1
- Design
- Implementierung
- Auswertung
- Zusammenfassung

Zusammenfassung

- gemeinsam genutzte Adressräume über Prozessorgrenzen hinweg
  - reduziert die Menge der für eine Arbeitslast benötigten Tasks
  - hoher Speicherverbrauch durch Duplikation von Kern-Datenstrukturen entfällt
  - Adressraummanipulationen sind fair und echtzeitkompatibel
- geeignet für eine hypothetische L4LinuxMP-Implementierung
  - Duplikation der Service-Threads im Linux-Server
  - ein L4-Thread für jede Linux-Task auf jedem Prozessor
  - Thread-Migration mit Hilfe eines Kern-Mechanismus nicht notwendig

TECHNISCHE UNIVERSITÄT DRESDEN

Ausblick

- unterbrechbares Message-Box-System
- IPI-Ratenbegrenzung
- globale Warteliste der MP-Preemption-Locks durch prozessorlokale Listen ersetzen
  - bessere Abschätzbarkeit der Zeit bis zur Lock-Zuweisung
- Thread-Migration durch Kern-Mechanismus
  - Anpassung des Message-Box-Systems
  - Anpassung der Scheduling-Parameter nach der Migration

TECHNISCHE UNIVERSITÄT DRESDEN

Fragen