Anomalien in Mehrkern-Systemen

Maximilian Wagner

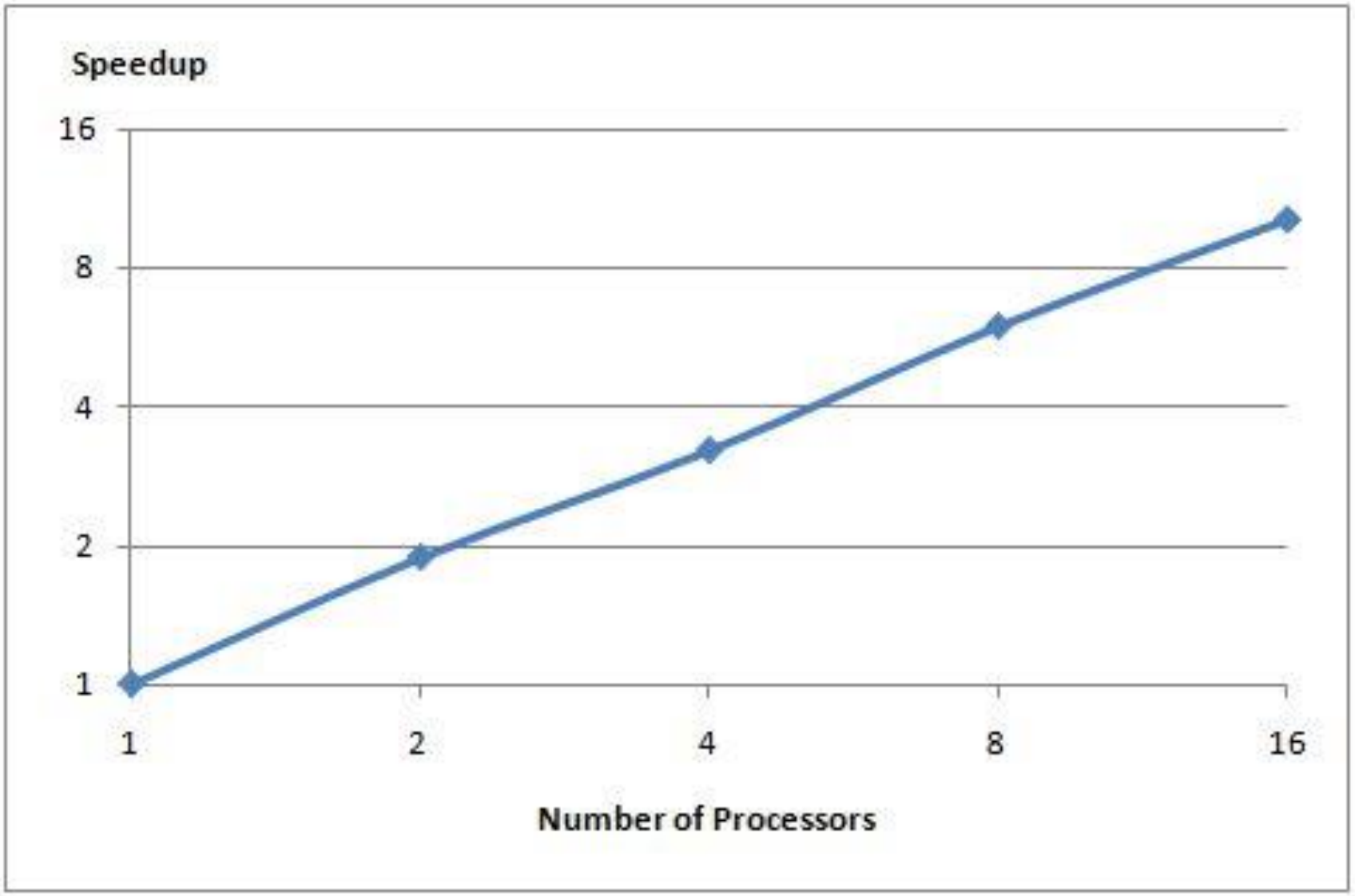
Warum läuft das nicht schneller?

LEHRSTUHL FÜR VERTEILTE SYSTEME
UND BETRIEBSSYSTEME

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Speedup bei steigender Anzahl von Prozessoren

(Quelle: https://software.intel.com/en-us/articles/multicore-enabling-a-binary-decision-diagram-algorithm)



Aber?

Also: mehr Prozessoren = höhere Geschwindigkeit?

Leider nicht immer!

- Fall 1: Geschwindigkeit bleibt gleich.
  - Möglicher Grund: Alle Aufgaben verteilt, zusätzliche Prozessoren bringen nichts. Aber sie schaden auch nicht.
- Fall 2: Geschwindigkeit sinkt!
  - Möglicher Grund: ?

Was sind Anomalien?

- Eine Anomalie ist (laut Duden)
  1. Abweichung vom Normalen; Abnormität
  2. Unregelmäßiges Verhalten im Vergleich zu Anderen

- Man könnte auch sagen:

  „Eine Anomalie ist eine Abweichung von erwartetem Verhalten.“

- Welches Verhalten erwarten wir?
  - Betrachte ein reales System
  - Erstelle eine Abstraktion
  - Führe auf dieser eine Zeitanalyse aus (*BCET/WCET*)
  - Alle zukünftigen Ausführungszeiten liegen innerhalb [*BCET;WCET*]

- Was, wenn einzelne Fälle aber außerhalb des Intervalls liegen? => Anomalie!

Gliederung

~~I. Einleitung~~

II. Abstraktionen:
   a) Was ist das?
   b) Wichtige Eigenschaften

III. Interferenzen:
   a) Inhärent
   b) Virtuell

IV. Beispiele für Anomalien:
   a) Zeitlich
   b) Domino-Effekte

V. Geteilte Betriebsmittel:
   a) Pipelines,
   b) Caches
   c) Busse

VI. Ausblick

Abstraktionen: *Was ist das?*

- Eine Abstraktion:
  - Ist ein versimpeltes Modell eines realen Systems
  - Dient zur Analyse dieses Systems
  - Für uns vor allem: Zeitanalyse (*BCET/WCET*)
  - Im Rahmen des Vortrags:
    - Ist wie ein Zustandsautomat aufgebaut, d.h.:
      - Wir haben Anfangs-, Zwischen- und Endzustände
      - Jeder Zustand hat einen oder mehrere Folgezustände
  - Mit Hilfe einer Abstraktion ermitteln wir also unser zu erwartendes Intervall: [*BCET;WCET*]

- Zusammenhang zu Anomalien:
  - Wir beobachten Einzelfälle außerhalb des Intervalls
  - Unser Intervall ist falsch
  - Unsere Analyse war fehlerhaft!

Abstraktionen: *Wichtige Eigenschaften*

- Wie vermeiden wir eine fehlerhafte Analyse?
  - Möglichst simple Abstraktion
- Denn: eine simple Abstraktion ist leicht zu analysieren
  - Sehr hohe Wahrscheinlichkeit, dass unser Intervall korrekt ist
  - Geringe Wahrscheinlichkeit für Anomalien
- Ideale Eigenschaften einer simplen Abstraktion:
  - Determinismus
  - Nur ein Anfangszustand
- Mehrkern-Systeme oft problematisch zu Abstrahieren:
  - Betriebsmittel werden geteilt und interagieren miteinander
  - Es treten Interferenzen auf

Gliederung

~~I. Einleitung~~

~~II. Abstraktionen:~~

~~a) Was ist das?~~

~~b) Wichtige Eigenschaften~~

III. Interferenzen:

a) Inhärent

b) Virtuell

IV. Beispiele für Anomalien:

a) Zeitlich

b) Domino-Effekte

V. Geteilte Betriebsmittel:

a) Pipelines,

b) Caches

c) Busse

VI. Ausblick

Interferenzen

- Was sind Interferenzen?
  - Unerwünschte Interaktionen/Abhängigkeiten zwischen einzelnen Anwendungen
- Wodurch entstehen sie?
  - Durch die gemeinsame Verwendung von Betriebsmitteln
- Weshalb sind sie für uns problematisch?
  - Sie erzeugen Nicht-Determinismus. Dieser führt zu:
    - Einer komplexeren Abstraktion
    - Einer höheren Wahrscheinlichkeit einer fehlerhaften Analyse
- Welche Arten gibt es?
  - Inhärente und Virtuelle Interferenzen

Inhärente Interferenzen

- Was sind inhärente Interferenzen?
  - Treten in realen Systemen auf
  - Zugriffe auf geteilte Betriebsmittel
    - Durch andere Anwendungen
    - Zu unvorhersehbaren Zeitpunkten
    - Die unsere Anwendung beeinflussen
- Beispiel: geteilter Cache
  - Andere Anwendung benutzt unseren Cache mit
  - Wir haben statt Cache-Hit nun Cache-Miss
  - Unsere Ausführungszeit verändert sich
- Problem: erschweren Zeitanalyse
  - Zugriffe zu unvorhersehbaren Zeitpunkten schwierig zu analysieren
  - Dadurch wird unsere Analyse aber evtl. fehlerhaft!

Virtuelle Interferenzen

- Was sind virtuelle Interferenzen?
  - Treten nur in Abstraktionen auf
  - Grund: große Anzahl zu beachtender Zustände
    - Wir können nicht alle einzeln analysieren
    - Wir versuchen zu verallgemeinern
    - Eventuell verlieren wir entscheidende Information über Nachfolgerzustände
    - Deterministisches System => nicht-deterministische Abstraktion
- Bsp.: Anfangszustand eines Caches in laufendem System
  - Sehr viele Möglichkeiten
  - Unmöglich alle zu berücksichtigen
  - Verallgemeinern auf einen Anfangszustand: leerer Cache
- Problem: Nicht-Determinismus führt evtl. zu inkorrekter Analyse

Gliederung

~~I. Einleitung~~

~~II. Abstraktionen:~~

~~a) Was ist das?~~

~~b) Wichtige Eigenschaften~~

~~III. Interferenzen:~~

~~a) Inhärent~~

~~b) Virtuell~~

IV. Beispiele für Anomalien:

a) Zeitlich

b) Domino-Effekte

V. Geteilte Betriebsmittel:

a) Pipelines,

b) Caches

c) Busse

VI. Ausblick

Zeitliche Anomalie

- Definition:

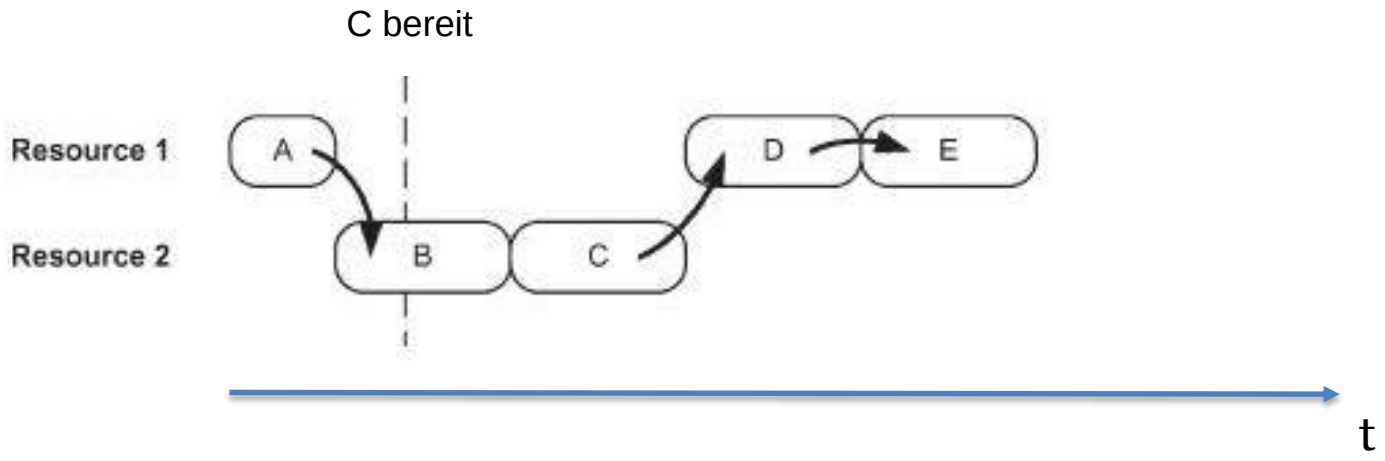
  „*Unterschiede in lokalem Zeitverhalten wirken sich unerwartet auf das globale Zeitverhalten aus.*“

- Also z.B.:
  - Lokal langsamer => global gleich schnell/schneller
  - Lokal schneller => global gleich schnell/langsamer

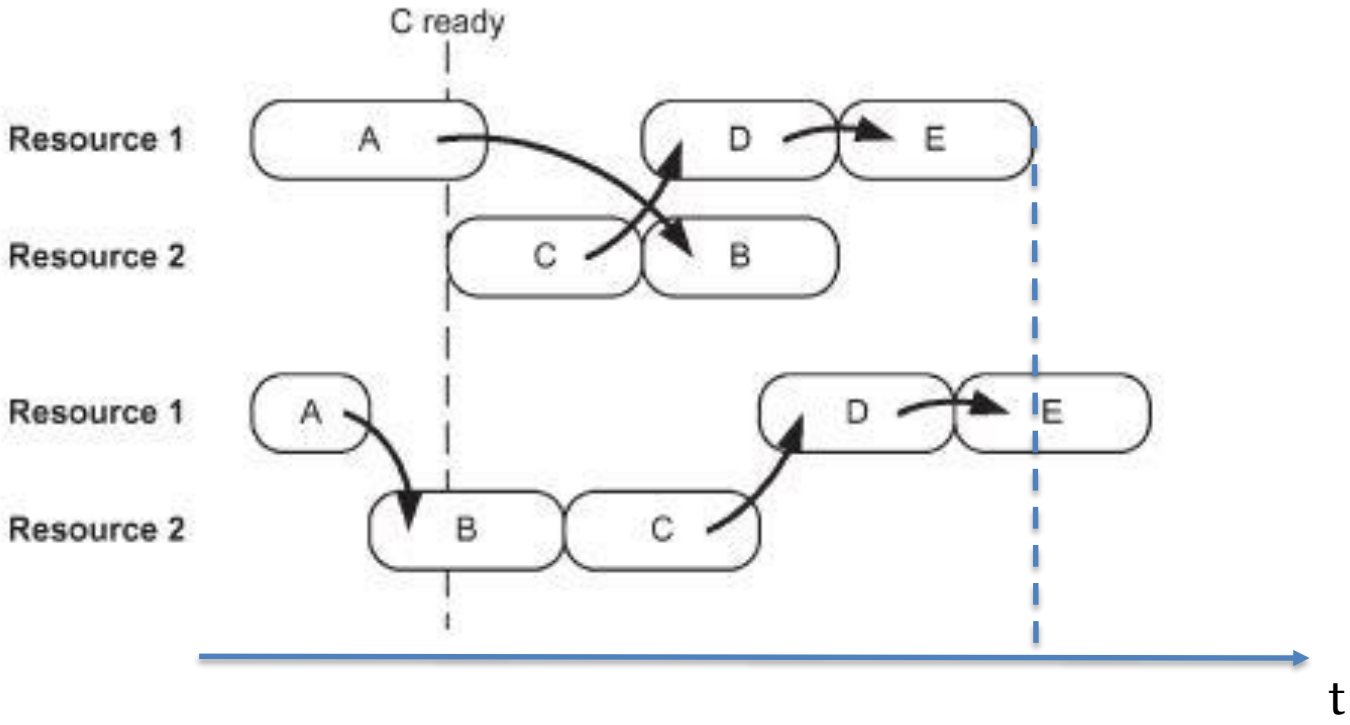
- Daher ist eine korrekte Zeitanalyse problematisch
  - Simples Aufaddieren lokaler BCET reicht nicht.
  - Interferenzen müssen berücksichtigt werden!
  - Das System muss als Ganzes analysiert werden.

Zeitliche Anomalie: Beispiel

- Annahmen:
  - 5 Anweisungen: *A-E*
  - Sequenzen: *A->B*, *C->D->E*
  - *C* zu Beginn nicht bereit
  - Prozessor 1: *A, D, E*
  - Prozessor 2: *B, C*



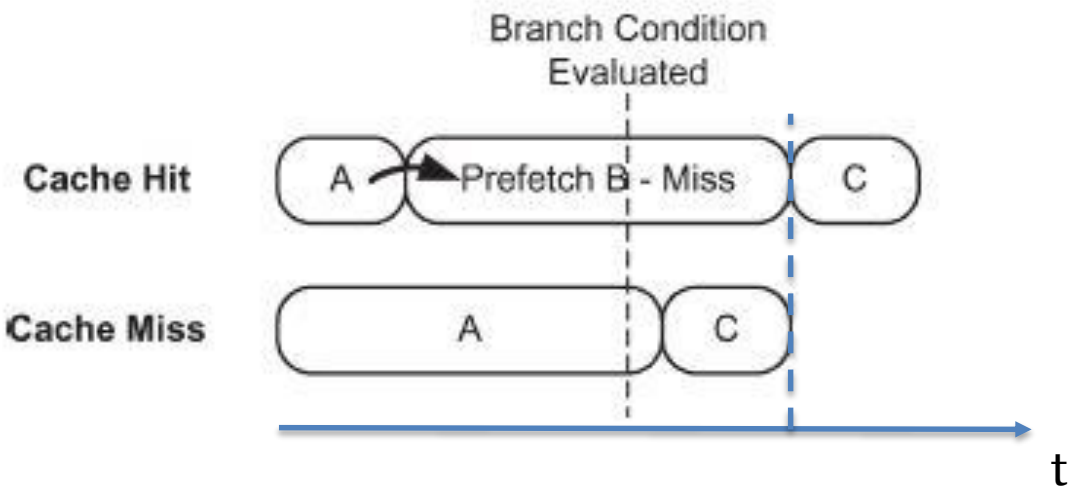
Zeitliche Anomalie: Beispiel 1



- *A* langsamer => Global schneller => zeitliche Anomalie

Zeitliche Anomalie: Beispiel 2

- Verzweigungsvorhersage:
  - *A* kann zu *B* abzweigen
  - Zu Beginn nicht evaluiert => Vorhersage



a) Falsche Vorhersage => unnötiges Vorladen von *B*
b) Durch Vorladen von *B* wird Cache beeinflusst => veränderter Cachezustand
c) Veränderter Cache kann neue Cache-Fehler verursachen
d) Global langsamere Ausführungszeit => zeitliche Anomalie

Domino-Effekt

- Was ist ein Domino-Effekt?
  - Eine Anwendung wird wiederholt ausgeführt
  - Man betrachtet zwei Anfangszustände: *s, t*
    - Anfangszustand s: die Ausführungszeit bleibt konstant
    - Anfangszustand t: die Ausführungszeit steigt mit jeder Iteration
  - T(t) – T(s) steigt also auch mit jeder Iteration
  - Zeitdifferenz ist also nicht konstant

- Wo liegt das Problem?
  - Die Zeitdifferenz lässt sich nicht vorhersagen
  - Konkrete Analyse der Iterationsanzahl notwendig
  - Alle Anfangszustände müssen betrachtet werden
  - Komplexere Abstraktion

Gliederung

~~I. Einleitung~~

~~II. Abstraktionen:~~

~~a) Was ist das?~~

~~b) Wichtige Eigenschaften~~

~~III. Interferenzen:~~

~~a) Inhärent~~

~~b) Virtuell~~

~~IV. Beispiele für Anomalien:~~

~~a) Zeitlich~~

~~b) Domino-Effekte~~

V. Geteilte Betriebsmittel:

a) Pipelines,

b) Caches

c) Busse

VI. Ausblick

Pipelines

- Wie funktionieren Pipelines?
  - Sequentielle Anwendungsblöcke werden teil-parallel ausgeführt
- Probleme?
  - Zugriffe auf z.B. gemeinsamen Speicher nun überlappend
  - Neue inhärente Interferenzen
  - Mehr mögliche Zustände
  - Komplexere Abstraktion nötig
  - Höhere Wahrscheinlichkeit, nicht alle Zustände betrachten zu können
  - Neue virtuelle Interferenzen
- Beide Arten von Interferenzen können also durch Pipelines verursacht werden.

Pipelines

- Zudem: Je mehr Funktionen die Pipeline hat, desto höher die Wahrscheinlichkeit für Interferenzen.
  - Funktionen: z.B. Superskalarität, Out-of-Order-Vorgehen, Anzahl Stufen
- Diese Funktionen sind jedoch idR. leistungssteigernd.

- Fazit:
  - Pipelines bringen Leistung, aber auch Interferenzen.
  - Komplexere Pipeline = mehr Interferenzen.

Gliederung

~~I. Einleitung~~

~~II. Abstraktionen:~~

~~a) Was ist das?~~

~~b) Wichtige Eigenschaften~~

~~III. Interferenzen:~~

~~a) Inhärent~~

~~b) Virtuell~~

~~IV. Beispiele für Anomalien:~~

~~a) Zeitlich~~

~~b) Domino-Effekte~~

V. Geteilte Betriebsmittel:

~~a) Pipelines,~~

b) Caches

c) Busse

VI. Ausblick

Caches

- Was ist ein Cache?
  - Prozessornaher Zwischenspeicher
  - Schneller lesbar als Hauptspeicher

- Wie funktioniert ein Cache?
  - Inhalt aus Hauptspeicher wird gelesen und im Cache zwischengespeichert
  - Bei erneuter Verwendung des Inhalts wird geprüft, ob er sich noch im Cache befindet
    - Falls ja: Cache-Hit
    - Falls nein: Cache-Miss

Caches

- Problematik:
  - Bereits einzelne Cache-Fehler kosten sehr viel Zeit
  - Große Zeitdifferenz zwischen Treffer und Fehler
  - Bei Unklarheit müssen beide Fälle berücksichtigt werden
  - BCET/WCET werden ungenauer

- Vorhersagbarkeit von Treffer/Fehler
  - Hängt von Ersetzungsstrategie ab
  - Strategien mit guter Vorhersagbarkeit:
    - Least-Recently-Used (LRU)
    - First-in-First-Out (FIFO)
    - Andere extrem schlecht vorhersehbar => ungeeignet

- Strategie mit sehr guter Vorhersagbarkeit: LRU
  - Leicht bestimmbar, ob Speicherzugriff Cache-Treffer/Fehler ist
  - Wichtig, da im Zweifelsfall beide berücksichtig werden müssen (zeitliche Anomalien)

Caches

- Auch relevant: Anfangszustand des Caches
  - Leider schwer konkret zu ermitteln, da viele Möglichkeiten vorhanden sind
  - Korrekte Analyse müsste alle berücksichtigen

- Glücklicherweise:
  - LRU ist nahezu unabhängig vom Anfangszustand
  - Vor allem im Vergleich zu FIFO

- Fazit:
  - Vorhersagen ob Cache-Treffer/-Fehler eintreten wird ist wichtig
  - Vorhersagbarkeit hängt von Ersetzungsstrategie ab
  - Strategie mit guter Vorhersagbarkeit: LRU

Gliederung

~~I. Einleitung~~

~~II. Abstraktionen:~~

~~a) Was ist das?~~

~~b) Wichtige Eigenschaften~~

~~III. Interferenzen:~~

~~a) Inhärent~~

~~b) Virtuell~~

~~IV. Beispiele für Anomalien:~~

~~a) Zeitlich~~

~~b) Domino-Effekte~~

V. Geteilte Betriebsmittel:

~~a) Pipelines,~~

~~b) Caches~~

c) Busse

VI. Ausblick

Busse

- Was sind Busse?
  - Verbindung zwischen mehreren Objekten
    - Bspw. zwischen Prozessoren und Arbeitsspeicher
  - Langsamer getaktet als Prozessoren
  - Bit-Seriell oder parallel (eine oder mehrere Leitungen)

- Problem 1: langsamer Takt
  - Prozessoren müssen idR. auf Busse warten
  - Anzahl an Zyklen, die maximal gewartet werden muss, ist fest
  - Diese berechnet sich über:

$$\#max.Zyklen := \frac{f_{Prozessor}}{ggT(f_{Prozessor}, f_{Bus})}$$

  - Weniger Zyklen = weniger mögliche Folgezustände in Abstraktion
  - ➢ Simplere Abstraktion
  - Guter Kompromiss: $f_{Bus} = \frac{f_{Prozessor}}{2}$ => $\#max.Zyklen = 2$

Busse

- Problem 2: parallele Busse
  - Bit-serielle Busse sind unproblematisch, aber langsamer
  - Parallele Busse erlauben Bus-Pipelining
    - Gleichzeitiges Übertragen mehrerer Datenstränge
    - Selbe Interferenz-Probleme wie bei normalen Pipelines
    - Erfordert Koordination
    - Koordination erfolgt über verschiedene Strategien

- Strategie mit guter Vorhersehbarkeit: *zentraler Bus-Arbiter*
  - Teilnehmer stellen Anfragen
  - Anfragen dürfen nur von *Bus-Meistern* (idR. alle Kerne) gestellt werden
  - *Arbiter* gewährt Zugriff nach konkreter Logik
  - Bsp.: *Zeitmultiplex-Verfahren* (*TDMA*):
    - Jeder Meister bekommt feste Zeitfenster zugeteilt
  - Großer Vorteil: Zugriffsverteilung ist strikt deterministisch!

Busse

- Fazit:
  - Schnell getaktete Busse sind besser
  - Bei parallelen Bussen ist deterministische Strategie vorteilhaft

- Strategie mit guter Vorhersehbarkeit: *zentraler Bus-Arbiter*
  - Teilnehmer stellen Anfragen
  - Anfragen dürfen nur von *Bus-Meistern* (idR. alle Kerne) gestellt werden
  - *Arbiter* gewährt Zugriff nach konkreter Logik
  - Bsp.: *Zeitmultiplex-Verfahren* (*TDMA*):
    - Jeder Meister bekommt feste Zeitfenster zugeteilt
  - Großer Vorteil: Zugriffsverteilung ist strikt deterministisch!

Gliederung

~~I. Einleitung~~

~~II. Abstraktionen:~~

~~a) Was ist das?~~

~~b) Wichtige Eigenschaften~~

~~III. Interferenzen:~~

~~a) Inhärent~~

~~b) Virtuell~~

~~IV. Beispiele für Anomalien:~~

~~a) Zeitlich~~

~~b) Domino-Effekte~~

~~V. Geteilte Betriebsmittel:~~

~~a) Pipelines,~~

~~b) Caches~~

~~c) Busse~~

VI. Ausblick

Was haben wir gelernt?

- Abstraktionen ergeben BCET/WCET-Intervall eines Systems
- Wird das Intervall verletzt, liegt eine Anomalie vor.
- In der Regel war somit unsere Abstraktion nicht korrekt/vollständig.

- Es gibt inhärente/virtuelle Interferenzen. Beide sind schlecht.
- Hauptgrund: geteilte Betriebsmittel.

- Zeitliche Anomalien und Domino-Effekte sind zu beachten.

- Pipelines, Caches und Busse steigern die Leistung.
- Aber sie bringen Interferenzen und andere Probleme mit sich.
- Diese machen die Zeitanalyse schwieriger.
- Somit steigt die Chance einer inkorrekten Zeitanalyse.

Was heißt das für Mehrkern-Echtzeitsysteme?

- Konflikt: mehr Leistung vs. Anomalie-Freiheit
  - Geteilte Verwendung von Betriebsmitteln
  - Komplexere Betriebsmittel
  - Interaktionen zwischen Betriebsmitteln
- Idealerweise verzichtet man komplett auf geteilte Betriebsmittel
  - IdR. jedoch nicht machbar
  - Für Mehrkern-Echtzeitsysteme muss also eine gute Balance gefunden werden

"The principle to be applied in the design of multicore architecture with predictable timing behavior is the systematic elimination of interference on shared resources wherever they are not absolutely needed for performance"

[Wilhelm 2009, Mem. Hierarchies[…], IEEE-Journal]

Warum läuft das also nun nicht schneller?

- Wir wissen
  - Interferenzen sind problematisch
  - Inhärente haben wir im Code aber bereits berücksichtigt
  - Trotzdem noch nicht schneller?

- Naheliegende Antwort
  - Unsere Abstraktion war fehlerhaft
  - Wir haben virtuelle Interferenzen vergessen!

Fragen?