Echtzeitfähigkeit virtueller Maschinen

Robert Kaiser[1] Dieter Zöbel[2]

[1]Fachbereich 2 – Technik, Informatik und Wirtschaft
Fachhochschule Bingen

[2]Fachbereich 4 – Informatik
Universität Koblenz-Landau

Echtzeit 2010, Boppard, 18.11.2008

Inhalt

1. Einleitung
   - Motivation
   - Herausforderungen
2. Planung unter Virtualisierungsumgebungen
   - Hierarchischer Scheduler
   - Planbarkeit
3. Kosten der Virtualisierung
   - Planungsbedingte Verluste
   - Umschaltkosten
4. Zusammenfassung

Inhalt

1. Einleitung
   - Motivation
   - Herausforderungen
2. Planung unter Virtualisierungsumgebungen
   - Hierarchischer Scheduler
   - Planbarkeit
3. Kosten der Virtualisierung
   - Planungsbedingte Verluste
   - Umschaltkosten
4. Zusammenfassung

Motivation

Rechenleistung eingebetteter Systeme steigt überproportional gegenüber Kosten

- $\longrightarrow$ Potenzial ...
  - ... **wirtschaftlich:** höhere Rechenleistung bei geringeren Kosten
  - ... **technisch:** weniger Bauteile $\Rightarrow$ geringere Ausfallwahrscheinlichkeit
- $\Rightarrow$ Ziel: Mehr Funktionen auf einem Rechner zusammenfassen
- $\Rightarrow$ Insgesamt weniger, dafür leistungsfähigere Rechner

$\Rightarrow$ *Konsolidierung* eingebetteter Systeme steht an

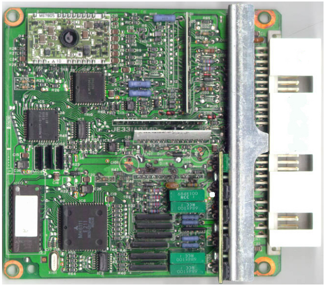


Abbildung: KFZ-Steuergerät

Motivation

Rechenleistung eingebetteter Systeme steigt überproportional gegenüber Kosten

→ Potenzial ...

... **wirtschaftlich:** höhere Rechenleistung bei geringeren Kosten

... **technisch:** weniger Bauteile ⇒ geringere Ausfallwahrscheinlichkeit

⇒ Ziel: Mehr Funktionen auf einem Rechner zusammenfassen

⇒ Insgesamt weniger, dafür leistungsfähigere Rechner

⇒ *Konsolidierung* eingebetteter Systeme steht an

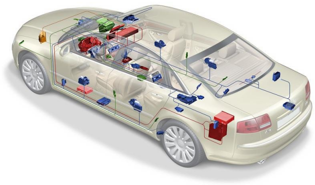


Abbildung: Verteilung von Steuergeräten in einem PKW

Motivation

Rechenleistung eingebetteter Systeme steigt überproportional gegenüber Kosten

$\longrightarrow$ Potenzial ...

... **wirtschaftlich:** höhere Rechenleistung bei geringeren Kosten

... **technisch:** weniger Bauteile $\Rightarrow$ geringere Ausfallwahrscheinlichkeit

$\Rightarrow$ Ziel: Mehr Funktionen auf einem Rechner zusammenfassen

$\Rightarrow$ Insgesamt weniger, dafür leistungsfähigere Rechner

$\Rightarrow$ *Konsolidierung* eingebetteter Systeme steht an

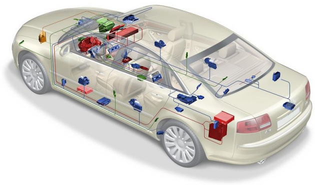


Abbildung: Verteilung von Steuergeräten in einem PKW

Motivation

Rechenleistung eingebetteter Systeme steigt überproportional gegenüber Kosten

- → Potenzial ...
  - ... **wirtschaftlich:** höhere Rechenleistung bei geringeren Kosten
  - ... **technisch:** weniger Bauteile ⇒ geringere Ausfallwahrscheinlichkeit
- ⇒ Ziel: Mehr Funktionen auf einem Rechner zusammenfassen
- ⇒ Insgesamt weniger, dafür leistungsfähigere Rechner

⇒ *Konsolidierung* eingebetteter Systeme steht an

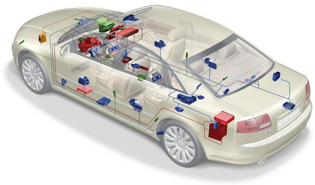


Abbildung: Verteilung von Steuergeräten in einem PKW

Konsolidierung eingebetter Systeme

Herausforderungen:

- Ausschließen ungewollter Wechselwirkungen
  („natürliche" Trennung der Subsysteme aufrecht erhalten)
- Verschiedene Laufzeitumgebungen auf einem Rechnerknoten
  (Jeder Anwendung ihre „eigene Welt")

Ähnlich den Anforderungen bei der Serverkonsolidierung

- Mehrere virtuelle Maschinen auf einem physischen Rechnerknoten
- Gegenseitige Isolierung, Aufteilung der Betriebsmittel

Aber: ist Virtualisierung „Echtzeitfähig"?

Konsolidierung eingebetter Systeme

Herausforderungen:

- Ausschließen ungewollter Wechselwirkungen
  („natürliche” Trennung der Subsysteme aufrecht erhalten)
- Verschiedene Laufzeitumgebungen auf einem Rechnerknoten
  (Jeder Anwendung ihre „eigene Welt”)

Ähnlich den Anforderungen bei der Serverkonsolidierung

- Mehrere virtuelle Maschinen auf einem physischen Rechnerknoten
- Gegenseitige Isolierung, Aufteilung der Betriebsmittel

Aber: ist Virtualisierung „Echtzeitfähig”?

Inhalt

1. Einleitung
   - Motivation
   - Herausforderungen
2. Planung unter Virtualisierungsumgebungen
   - Hierarchischer Scheduler
   - Planbarkeit
3. Kosten der Virtualisierung
   - Planungsbedingte Verluste
   - Umschaltkosten
4. Zusammenfassung

Hierarchischer Scheduler

Bei bisherigen Virtualisierungsansätzen:

- Einfluss der Virtualisierung auf das Zeitverhalten wurde nicht betrachtet (vgl. [Popek/Goldberg])

Für eingebettete Systeme nicht akzeptabel

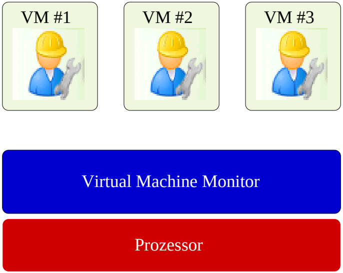
- Prozessor muss im Zeitmultiplex-Verfahren zugeteilt werden

⇒ VMM agiert als *globaler* Scheduler

- Virtuelle Maschinen enthalten i.A. Betriebssysteme mit Anwendungen

⇒ Betriebssysteme agieren als *lokale* Scheduler

Beide Scheduler-Ebenen sind zu betrachten



Hierarchischer Scheduler

Bei bisherigen Virtualisierungsansätzen:

- Einfluss der Virtualisierung auf das Zeitverhalten wurde nicht betrachtet (vgl. [Popek/Goldberg])

Für eingebettete Systeme nicht akzeptabel

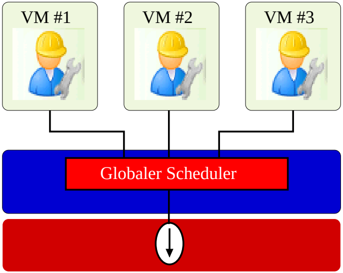
- Prozessor muss im Zeitmultiplex-Verfahren zugeteilt werden

⇒ VMM agiert als *globaler* Scheduler

- Virtuelle Maschinen enthalten i.A. Betriebssysteme mit Anwendungen

⇒ Betriebssysteme agieren als *lokale* Scheduler

Beide Scheduler-Ebenen sind zu betrachten



Hierarchischer Scheduler

Bei bisherigen Virtualisierungsansätzen:

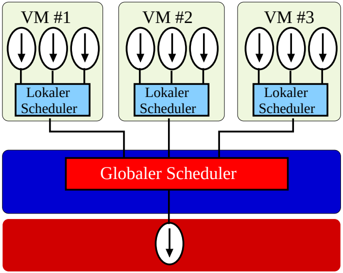
- Einfluss der Virtualisierung auf das Zeitverhalten wurde nicht betrachtet (vgl. [Popek/Goldberg])

Für eingebettete Systeme nicht akzeptabel

- Prozessor muss im Zeitmultiplex-Verfahren zugeteilt werden

⇒ VMM agiert als *globaler* Scheduler

- Virtuelle Maschinen enthalten i.A. Betriebssysteme mit Anwendungen



⇒ Betriebssysteme agieren als *lokale* Scheduler

Beide Scheduler-Ebenen sind zu betrachten

Hierarchischer Scheduler

Bei bisherigen Virtualisierungsansätzen:

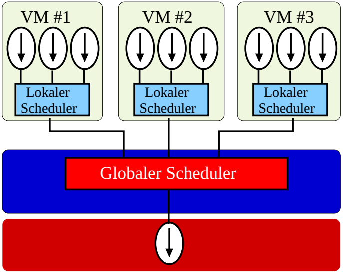
- Einfluss der Virtualisierung auf das Zeitverhalten wurde nicht betrachtet (vgl. [Popek/Goldberg])

Für eingebettete Systeme nicht akzeptabel

- Prozessor muss im Zeitmultiplex-Verfahren zugeteilt werden

⇒ VMM agiert als *globaler* Scheduler

- Virtuelle Maschinen enthalten i.A. Betriebssysteme mit Anwendungen



⇒ Betriebssysteme agieren als *lokale* Scheduler

Beide Scheduler-Ebenen sind zu betrachten

Hierarchischer Scheduler

Bei bisherigen Virtualisierungsansätzen:

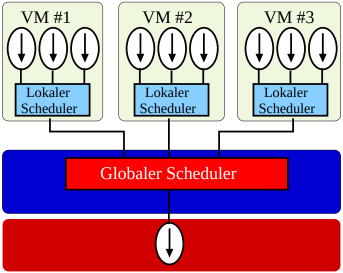
- Einfluss der Virtualisierung auf das Zeitverhalten wurde nicht betrachtet (vgl. [Popek/Goldberg])

Für eingebettete Systeme nicht akzeptabel

- Prozessor muss im Zeitmultiplex-Verfahren zugeteilt werden

⇒ VMM agiert als *globaler* Scheduler

- Virtuelle Maschinen enthalten i.A. Betriebssysteme mit Anwendungen

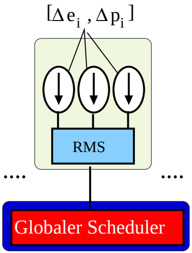


⇒ Betriebssysteme agieren als *lokale* Scheduler

Beide Scheduler-Ebenen sind zu betrachten

Prozessmodell nach Liu/Layland:

- **Gegeben:**
  1. $P = \{1, .., n\}$: Menge period. Prozesse mit:
     - Ausführunszeiten Δe_i
     - Periodendauern Δp_i
  2. Lokaler Scheduler (hier: RMS)

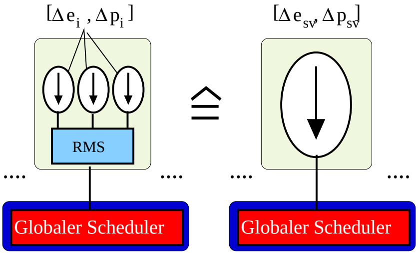


- **Gesucht:** Äquivalente Darstellung der Prozessmenge als ein periodischer *Stellvertreterprozess* mit Δe_{SV}, Δp_{SV}.

Stellvertreterprozess-Modell

Prozessmodell nach Liu/Layland:

- **Gegeben:**
  1. $P = \{1, .., n\}$: Menge period. Prozesse mit:
     - Ausführunszeiten Δe_i
     - Periodendauern Δp_i
  2. Lokaler Scheduler (hier: RMS)



- **Gesucht:** Äquivalente Darstellung der Prozessmenge als ein periodischer *Stellvertreterprozess* mit Δe_{sv}, Δp_{sv}.

Planen nach Monotonen Raten

- Vorschrift: Prioritäten umgekehrt proportional zu Periodendauern festlegen:

$$\Delta p_i > \Delta p_j \Leftrightarrow prio(i) < prio(j)$$

- Ansatz für Stellvertreterprozess:

1. Fasse alle **nicht** zugeteilten Zeiten als periodischen „Störprozess” δ auf.
2. Wenn $\Delta p_{sv} \leq min(\Delta p_i)$ kann δ als neuer höchstpriorer Prozess zu P hinzugefügt werden.

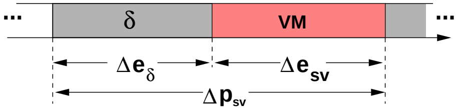
$\Rightarrow$ **RMS bleibt anwendbar**

Planen nach Monotonen Raten

- Vorschrift: Prioritäten umgekehrt proportional zu Periodendauern festlegen:

$$\Delta p_i > \Delta p_j \Leftrightarrow prio(i) < prio(j)$$

- Ansatz für Stellvertreterprozess:



1. Fasse alle **nicht** zugeteilten Zeiten als periodischen „Störprozess" δ auf.
2. Wenn $\Delta p_{sv} \leq min(\Delta p_i)$ kann δ als neuer höchstpriorer Prozess zu P hinzugefügt werden.

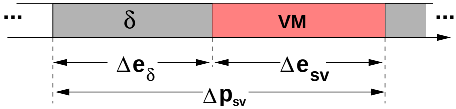
$\Rightarrow$ **RMS bleibt anwendbar**

Planen nach Monotonen Raten

- Vorschrift: Prioritäten umgekehrt proportional zu Periodendauern festlegen:

$$\Delta p_i > \Delta p_j \Leftrightarrow prio(i) < prio(j)$$

- Ansatz für Stellvertreterprozess:



1. Fasse alle **nicht** zugeteilten Zeiten als periodischen „Störprozess" δ auf.
2. Wenn $\Delta p_{sv} \leq min(\Delta p_i)$ kann δ als neuer höchstpriorer Prozess zu P hinzugefügt werden.

$\Rightarrow$ **RMS bleibt anwendbar**

Brauchbarkeit des Plans

Frage: Bis zu welcher Auslastung ist der Plan brauchbar?

- Abgewandelte Form des Beweises von Liu/Layland ($\longrightarrow$ [Kaiser/Zöbel ETFA09]). Ergebnis:

$$U(P)_{min} = n \cdot \left(\sqrt[n]{\frac{2}{U_\delta + 1}} - 1 \right)$$

$$\lim_{n \to \infty} U(P)_{min} = \ln \left(\frac{2}{U_\delta + 1} \right)$$

- Dabei ist: $U_\delta = \frac{\Delta e_\delta}{\Delta p_{sv}}$ (Auslastung durch den Störprozess δ)

Periodendauer und Ausführungszeit

- Periodendauer lt. Voraussetzung (s.o.): $\Delta p_{sv} = min(\Delta p_i)$
- Außerdem gilt (s.o.): $\Delta e_{sv} + \Delta e_{\delta} = \Delta p_{sv}$
  $\Rightarrow U_{sv} = \frac{\Delta e_{sv}}{\Delta p_{sv}} = 1 - U_{\delta}$

$$\Rightarrow \Delta e_{sv} = 2 \cdot \Delta p_{sv} \cdot \left(1 - \frac{1}{\left(\frac{U(P)}{n} + 1 \right)^n} \right)$$

$$\lim_{n \to \infty} \Rightarrow \Delta e_{sv} = 2 \cdot \Delta p_{sv} \cdot \left(1 - e^{-U(P)} \right)$$

$\Rightarrow$ Damit: $\Delta e_{sv}, \Delta p_{sv}$ für alle VMs berechenbar

$\Rightarrow$ Globaler Scheduler plant wiederum periodische Prozesse

$\Rightarrow$ Kann seinerseits z.B. RMS oder EDF einsetzen

Inhalt

1. Einleitung
   - Motivation
   - Herausforderungen
2. Planung unter Virtualisierungsumgebungen
   - Hierarchischer Scheduler
   - Planbarkeit
3. Kosten der Virtualisierung
   - Planungsbedingte Verluste
   - Umschaltkosten
4. Zusammenfassung

Planungsbedingte Verluste

Verluste sind unvermeidlich:

- Es gibt Zeitpunkte, zu denen eine VM (d.h. ihr Stellvertreterprozess) den Prozessor besitzt, aber kein lokaler Prozess ist rechenwillig
- D.h. mit:
  - $U_{sv} = \frac{\Delta e_{sv}}{\Delta p_{sv}}$ *Äußere Auslastung*
  - $U(P) = \sum_{i \in P} \frac{\Delta e_i}{\Delta p_i}$ *Innere Auslastung* durch P
- gilt: *relative Auslastung*: $u_{sv} = \frac{U_{sv}}{U(P)} > 1$

Maß für planungsbedingte Verluste (für große Prozessanzahlen):

$$u_{sv} = \frac{2}{U(P)} \cdot \left(1 - e^{-U(P)}\right)$$

Für kleine Einzel-Auslastungen $U(P)$ gilt $u_{sv} \longrightarrow 2$

$\Rightarrow$ Planungsbedingte Verluste sind beschränkt (auf maximal 50%)

Planungsbedingte Verluste

Verluste sind unvermeidlich:

- Es gibt Zeitpunkte, zu denen eine VM (d.h. ihr Stellvertreterprozess) den Prozessor besitzt, aber kein lokaler Prozess ist rechenwillig
- D.h. mit:
  - $U_{sv} = \frac{\Delta e_{sv}}{\Delta p_{sv}}$ *Äußere Auslastung*
  - $U(P) = \sum_{i \in P} \frac{\Delta e_i}{\Delta p_i}$ *Innere Auslastung* durch P
- gilt: *relative Auslastung*: $u_{sv} = \frac{U_{sv}}{U(P)} > 1$

Maß für planungsbedingte Verluste (für große Prozessanzahlen):

$$u_{sv} = \frac{2}{U(P)} \cdot \left(1 - e^{-U(P)}\right)$$

Für kleine Einzel-Auslastungen $U(P)$ gilt $u_{sv} \longrightarrow 2$

$\Rightarrow$ Planungsbedingte Verluste sind beschränkt (auf maximal 50%)

Grenzen der Prozesswechsel-Frequenz

In bisherigen Betrachtungen wurde Prozesswechsel-Aufwand (wie oft) vernachlässigt, aber:

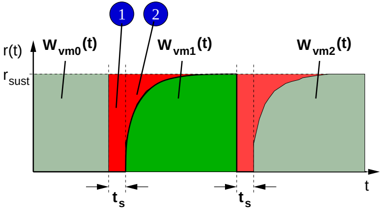
- Bedingung für Anwendbarkeit RMS (s.o.) kann zu hohen Umschaltfrequenzen führen
- „Dynamische Umnutzung" planungsbedingter Verluste prinzipiell möglich, aber je nach Umschaltfrequenz nicht sinnvoll

$\Rightarrow$ ***Quantitatives* Erfassen der Umschaltkosten ist nötig**

- Problem: Wg. Caches ist eine analytische Beschreibung dieser Kosten i.d.R. nicht möglich
  - Volle Kenntnis über Zustand und Verhalten der VMs wäre erforderlich
  - Selbst dann: Berechnung für on-line Scheduler zu komplex
- Ansatz hier: Empirische Vorgehensweise
  - Beobachten/Messen des Systemverhaltens
  - Approximation durch mit den Messwerten parametrisierte, funktionale Beschreibung

Umschaltverluste zusammengesetzt aus:

1. Unterbrechungszeit (fix)
2. Cache-bedingten Kosten (Zeitabhängig):



- Einfaches Modell des Cache-Verhaltens
- Worst-Case Verhalten ist reproduzierbar (und messbar)

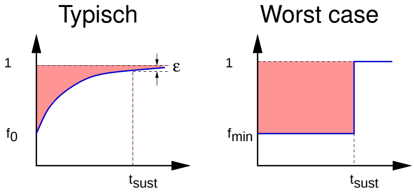
⇒ **Vorgehensweise:**

- Worst-Case Verhalten messen
- Geeignete Funktionen mit Messwerten parametrieren
- Verluste anhand Scheduler-Simulation aufsummieren

Modellieren der Umschaltverluste

Umschaltverluste zusammengesetzt aus:

1. Unterbrechungszeit (fix)
2. Cache-bedingten Kosten (Zeitabhängig):
   - Einfaches Modell des Cache-Verhaltens
   - Worst-Case Verhalten ist reproduzierbar (und messbar)



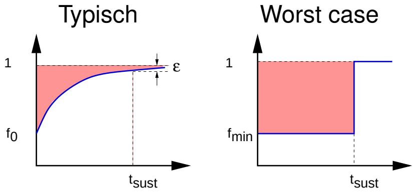
⇒ **Vorgehensweise:**

- Worst-Case Verhalten messen
- Geeignete Funktionen mit Messwerten parametrieren
- Verluste anhand Scheduler-Simulation aufsummieren

Modellieren der Umschaltverluste

Umschaltverluste zusammengesetzt aus:

1. Unterbrechungszeit (fix)
2. Cache-bedingten Kosten (Zeitabhängig):
   - Einfaches Modell des Cache-Verhaltens
   - Worst-Case Verhalten ist reproduzierbar (und messbar)



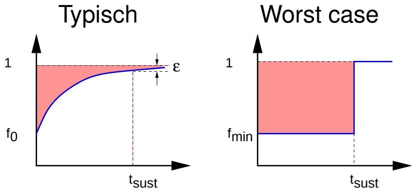
⇒ **Vorgehensweise:**

- Worst-Case Verhalten messen
- Geeignete Funktionen mit Messwerten parametrieren
- Verluste anhand Scheduler-Simulation aufsummieren

Modellieren der Umschaltverluste

Umschaltverluste zusammengesetzt aus:

1. Unterbrechungszeit (fix)
2. Cache-bedingten Kosten (Zeitabhängig):
   - Einfaches Modell des Cache-Verhaltens
   - Worst-Case Verhalten ist reproduzierbar (und messbar)



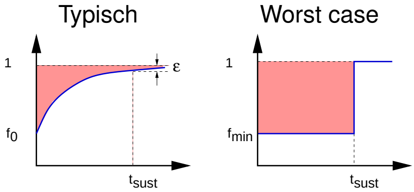
$\Rightarrow$ **Vorgehensweise:**

- Worst-Case Verhalten messen
- Geeignete Funktionen mit Messwerten parametrieren
- Verluste anhand Scheduler-Simulation aufsummieren

Modellieren der Umschaltverluste

Umschaltverluste zusammengesetzt aus:

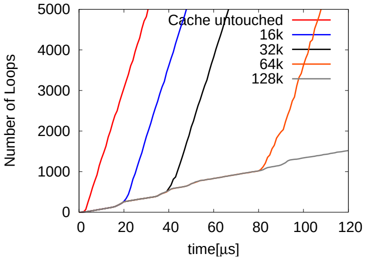
1. Unterbrechungszeit (fix)
2. Cache-bedingten Kosten (Zeitabhängig):
   - Einfaches Modell des Cache-Verhaltens
   - Worst-Case Verhalten ist reproduzierbar (und messbar)

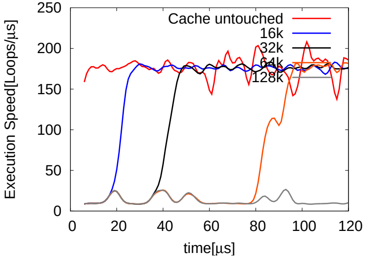


$\Rightarrow$ **Vorgehensweise:**

- Worst-Case Verhalten messen
- Geeignete Funktionen mit Messwerten parametrieren
- Verluste anhand Scheduler-Simulation aufsummieren

Messverfahren





- Gemessen: Arbeit (Schleifendurchläufe pro Zeiteinheit, verschiedene „working sets")

- Ableitung: Fortschrittsrate (= Rechenleistung)

Ergebnisse (1)

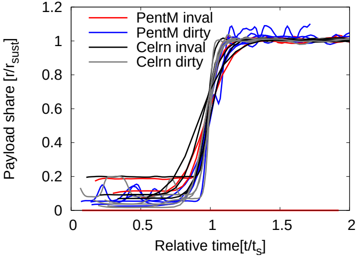
- Messergebnisse (i386):

Machine	WSS	Cache	f_0	t_{sust}
Cel.@2.5G	16k	dirty	0.11	22μs
Cel.@2.5G	32k	dirty	0.11	43μs
Cel.@2.5G	64k	dirty	0.09	85μs
Cel.@2.5G	16k	invd	0.17	12μs
Cel.@2.5G	32k	invd	0.18	20μs
Cel.@2.5G	64k	invd	0.20	35μs
PentM@1.5G	16k	dirty	0.09	25μs
PentM@1.5G	32k	dirty	0.08	38μs
PentM@1.5G	64k	dirty	0.12	91μs
PentM@1.5G	16k	invd	0.17	16μs
PentM@1.5G	32k	invd	0.12	29μs
PentM@1.5G	64k	invd	0.21	55μs

Testprozedur: Aufsteigende Cache-Zeilen
invd = Cache invalidated, dirty = cache flood-filled

- t_{sust} Wächst mit „working space”
- f_0 Zwischen (hier) 8% und 21%

- Normalisierte Werte:



⇒ Entspricht in guter Näherung dem erwarteten Verhalten

Ergebnisse (2)

- Messergebnisse (PowerPC[a]):
  - Single-level Cache, 16 KB

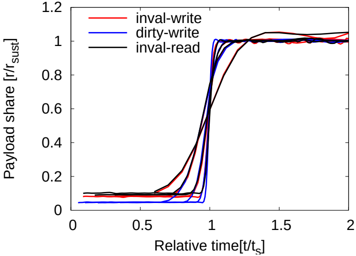
Testcase	WSS	Cache	f_0	t_{sust}
consec_wr	2k	dirty	0.05	$17\mu s$
consec_wr	4k	dirty	0.05	$31\mu s$
consec_wr	8k	dirty	0.05	$59\mu s$
consec_wr	16k	dirty	0.05	$116\mu s$
consec_wr	2k	invd	0.13	$10\mu s$
consec_wr	4k	invd	0.09	$19\mu s$
consec_wr	8k	invd	0.08	$35\mu s$
consec_wr	16k	invd	0.08	$69\mu s$
consec_rd	2k	invd	0.13	$10\mu s$
consec_rd	4k	invd	0.10	$19\mu s$
consec_rd	8k	invd	0.09	$35\mu s$
consec_rd	16k	invd	0.10	$68\mu s$

invd = Cache invalidated, dirty = cache flood-filled

- $t_s \sim WSS$
- f_0 zwischen 5% und 13%

  unabhängig von WSS

- Normalisierte Werte:



$\Rightarrow$ Entspricht Modellverhalten

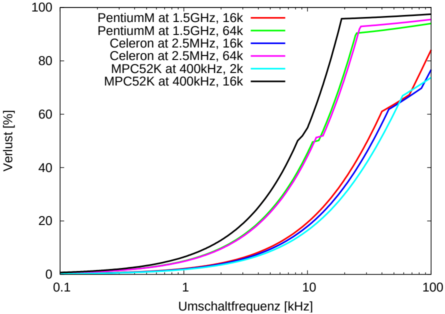
[a] MPC 5200 @ 400MHz

Ergebnisse (3)

Simulation eines Schedulers, parametriert mit ermittelten Werten

$\longrightarrow$ Worst-case-Umschaltverluste in Abhängigkeit von der Umschaltfrequenz

- Unterhalb ca. 1kHz: < 1%
  $\longrightarrow$ vernachlässigbar
- Darüber: Drastischer Anstieg
  (bis über 90%)



Faustregel: Zeitscheibendauern unter einer Millisekunde führen zu hohen Verlusten

- Verluste können Ausführungszeiten zugeschlagen werden

Inhalt

1. Einleitung
   - Motivation
   - Herausforderungen
2. Planung unter Virtualisierungsumgebungen
   - Hierarchischer Scheduler
   - Planbarkeit
3. Kosten der Virtualisierung
   - Planungsbedingte Verluste
   - Umschaltkosten
4. Zusammenfassung

Zusammenfassung

- Stellvertreterprozess-Modell zur Zusammenfassung von nach RMS priorisierten Prozessmengen
- Einfach anzuwendendes Kriterium zum Nachweis der Planbarkeit solcher Prozessmengen in virtuellen Maschinen
- Planungsbedinge Verluste...
  - \+ ... sind beschränkt
  - – ... können bis zu 50% betragen
  - $\longrightarrow$ Selbst dann: ökonomisch sinnvoll
- Modell zur approximation Cache-bedingter Umschaltkosten
  - \+ Bei Frequenzen unter 100 Hz $\longrightarrow$ vernachlässigbar
  - – Bei Frequenzen über ca 10kHz $\longrightarrow$ erheblich
  - $\longrightarrow$ Kompromiss unter Berücksichtigung der Anwendungsanforderungen

Das Ende...

Danke für Ihre Aufmerksamkeit!

... Fragen?