Koexistenz unterschiedlicher Zeitanforderungen in einem gemeinsamen Rechensystem

Robert Kaiser

robert.kaiser@sysgo.com

SYSGO AG

Fachhochschule Wiesbaden

Überblick

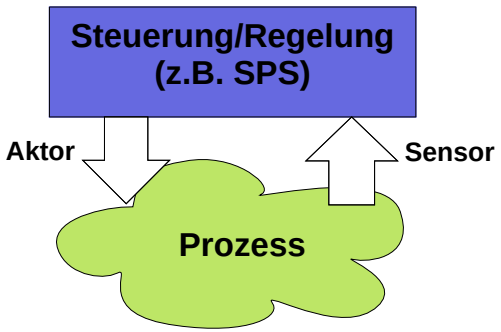
SYSGO
REAL-TIME SOLUTIONS

- Motivation
- Klassifikation der Rechenlasten
- Anforderungen an das Betriebssystem
- „Kern-im-Kern” Systeme und Virtualisierung
- Virtualisierung für Echtzeitsysteme
- Zusammenfassung/Ausblick

Motivation

SYSGO
REAL-TIME SOLUTIONS

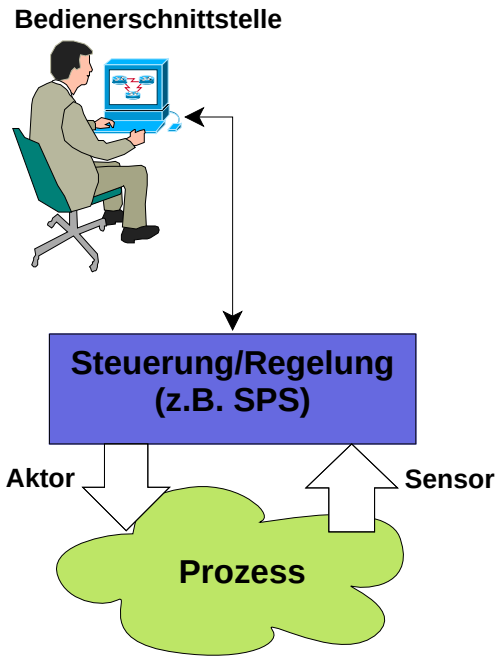
Alltägliche Echtzeitsysteme enthalten Komponenten mit unterschiedlichsten Zeitanforderungen, z.B.:



- Steuerung/Regelung:
  - zeitgesteuert („proaktiv”)
  - i.d.R. „harte” Echtzeit

Motivation

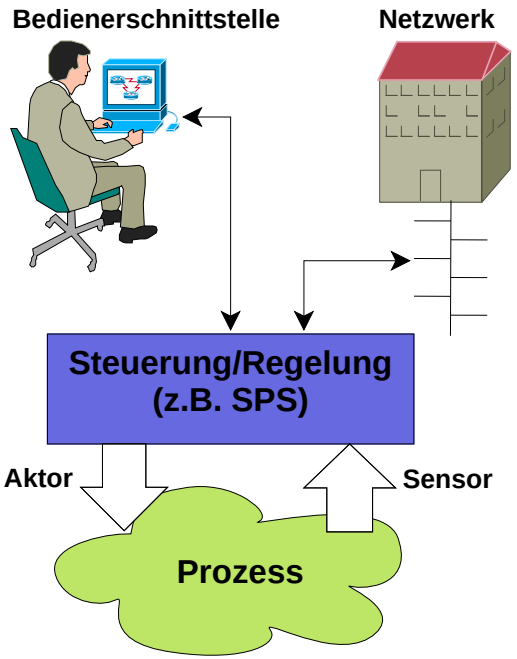
Alltägliche Echtzeitsysteme enthalten Komponenten mit unterschiedlichsten Zeitanforderungen, z.B.:



- Steuerung/Regelung:
  - zeitgesteuert („proaktiv”)
  - i.d.R. „harte” Echtzeit
- Anwenderschnittstelle:
  - ereignisgesteuert („reaktiv”)
  - i.d.R. „weiche” Echtzeit

Motivation

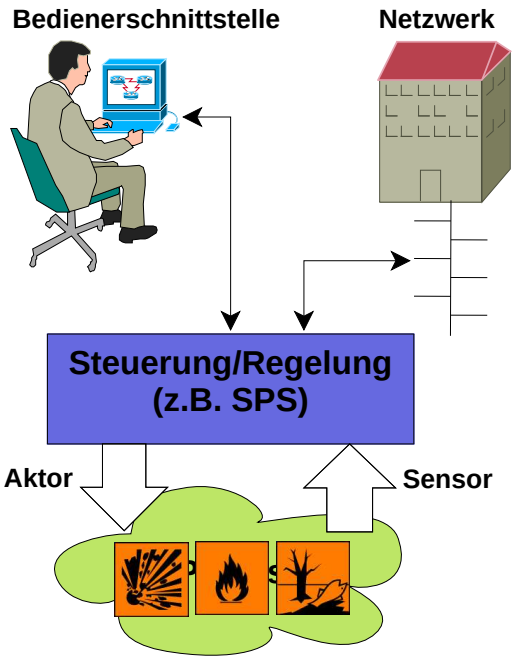
Alltägliche Echtzeitsysteme enthalten Komponenten mit unterschiedlichsten Zeitanforderungen, z.B.:



- Steuerung/Regelung:
  - zeitgesteuert („proaktiv”)
  - i.d.R. „harte” Echtzeit
- Anwenderschnittstelle:
  - ereignisgesteuert („reaktiv”)
  - i.d.R. „weiche” Echtzeit
- Netzwerkanbindung
  - sowohl „reaktiv” als auch „proaktiv”
  - keine (bzw. geringe) Zeitanforderungen

Motivation

Alltägliche Echtzeitsysteme enthalten Komponenten mit unterschiedlichsten Zeitanforderungen, z.B.:

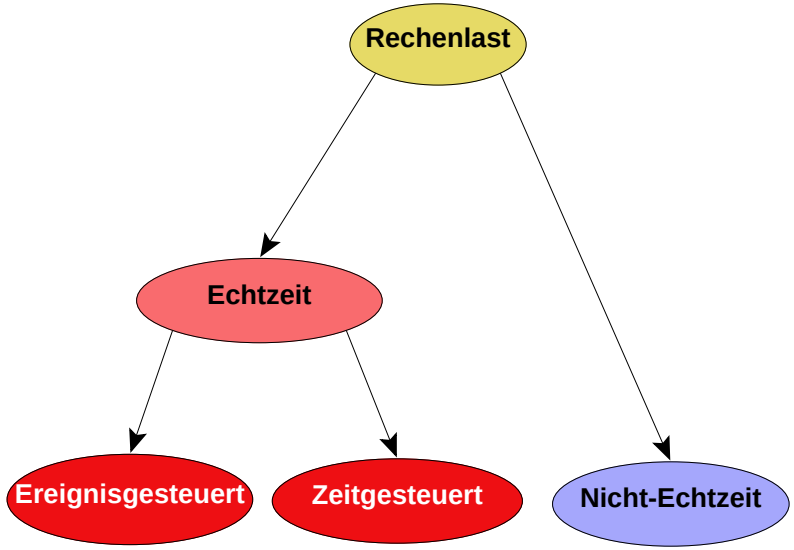


- Steuerung/Regelung:
  - zeitgesteuert („proaktiv”)
  - i.d.R. „harte” Echtzeit
- Anwenderschnittstelle:
  - ereignisgesteuert („reaktiv”)
  - i.d.R. „weiche” Echtzeit
- Netzwerkanbindung
  - sowohl „reaktiv” als auch „proaktiv”
  - keine (bzw. geringe) Zeitanforderungen
- häufig: sicherheitskritisch

Klassifikation der Rechenlasten

SYSGO REAL-TIME SOLUTIONS

- Nicht-Echtzeitanwendungen
  - keine Zeitgarantien
  - dynamisch wechselnder Betriebsmittelbedarf
  - „Best Effort”
- Echtzeitanwendungen
  - („harte” oder „weiche”) Zeitgarantien
  - gegebener (statischer) Betriebsmittelbedarf
  - dimensioniert unter worst-case Annahmen
  - Ziel: deterministisches Zeitverhalten
    - zeitgesteuert: absolut
    - ereignisgesteuert: bezogen auf Ereignisse

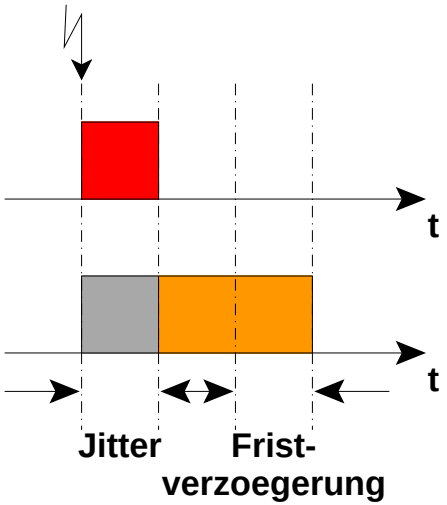


⇒ **gegensätzliche Konzepte, schwer miteinander vereinbar.**

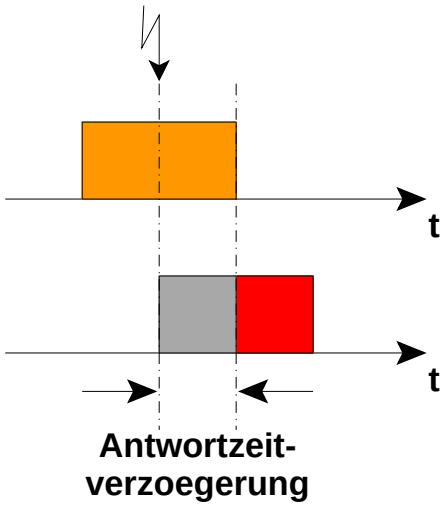
Zeit- vs. Ereignissteuerung (1)

Zeit- und Ereignissteuerung behindern sich gegenseitig:

Ereignissteuerung > Zeitsteuerung



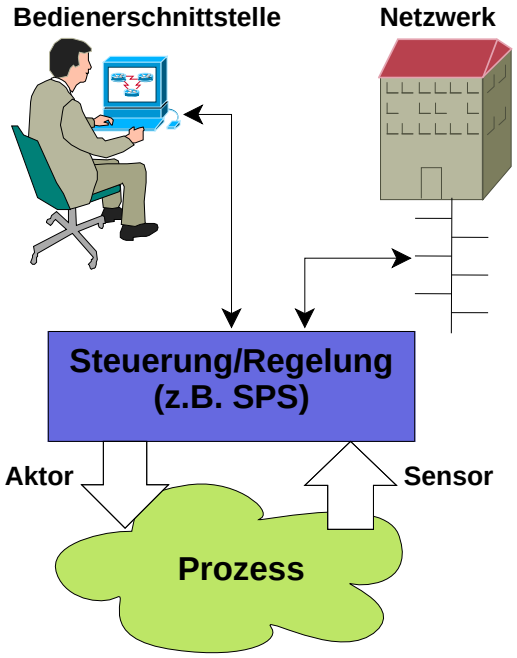
Zeitsteuerung > Ereignissteuerung



⇒ **Entwurfsentscheidung: welches hat Vorrang?**

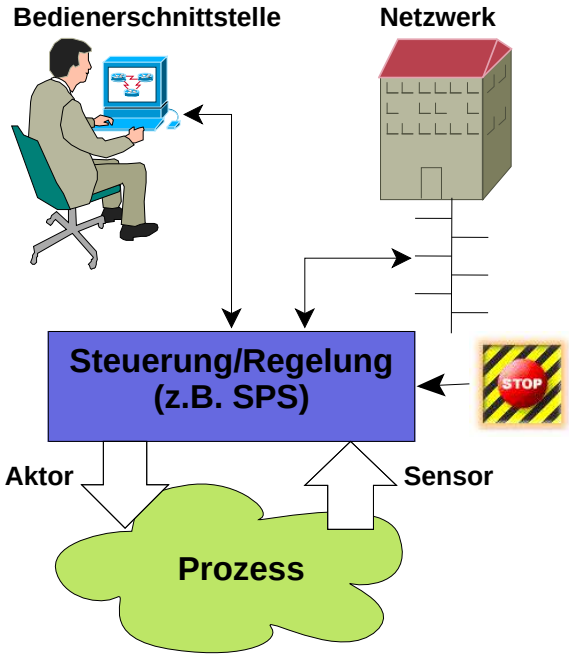
Zeit- vs. Ereignissteuerung (2)

- Welches Konzept hat Vorrang?
- Nicht immer einfach entscheidbar:



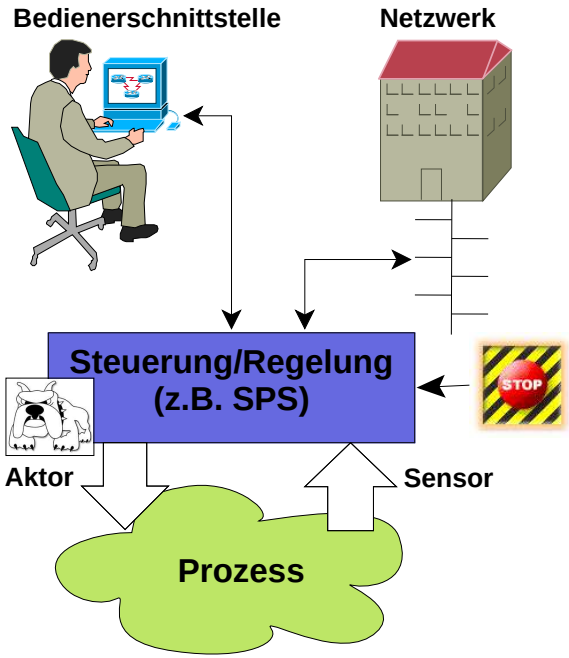
Zeit- vs. Ereignissteuerung (2)

- Welches Konzept hat Vorrang?
- Nicht immer einfach entscheidbar:
- Beispiel: Not-Aus Funktion
  - ereignisgetrieben
  - absoluter Vorrang



Zeit- vs. Ereignissteuerung (2)

- Welches Konzept hat Vorrang?
- Nicht immer einfach entscheidbar:
- Beispiel: Not-Aus Funktion
  - ereignisgetrieben
  - absoluter Vorrang
- Beispiel: Watchdog
  - zeitgetrieben
  - absoluter Vorrang



Auswirkung auf Betriebssysteme

SYSGO REAL-TIME SOLUTIONS

- Nicht-Echtzeitsystem
  - dynamische Betriebsmittelzuteilung
  - virtueller Speicher
  - Überbuchung
  - Ziel: Betriebsmittelauslastung
- Echtzeitsystem
  - deterministische (i.d.R. statische) Betriebsmittelzuteilung
  - Ziel: deterministisches Zeitverhalten
    - zeitgesteuert: wenig Jitter
    - ereignisgesteuert: schnelle Reaktion
- $\Rightarrow$ 3 weitgehend orthogonale Klassen von Funktionen
  1. Nicht-Echtzeit
  2. Echtzeit zeitgesteuert
  3. Echtzeit ereignisgesteuert

Auswirkung auf Betriebssysteme

SYSGO REAL-TIME SOLUTIONS

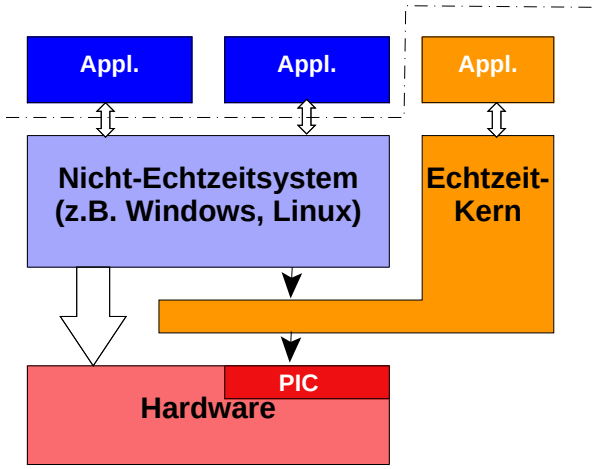
Klassisch: Betriebssystemschnittstelle ist global

- Alle Funktionen in einer Schnittstelle
  - + eine universelle Schnittstelle für alle Anwendungen
  - - zahlreiche Funktionen, die u.U. nur zum Teil genutzt werden
  - - Betriebssystem („trusted code”) wird sehr komplex
  - - Sicherheitsnachweis schwer zu führen
- Mehrere, spezialisierte Schnittstellen
  - + „bedarfsgerechte” Schnittstellenfunktionalität
  - + besser beherrschbar
  - + Möglichkeit, Standard-APIs und Speziallösungen zu kombinieren
  - - Kommunikation über Schnittstellengrenzen wird aufwändiger

Ansatz 1: „Kern-im-Kern” Systeme

SYSGO REAL-TIME SOLUTIONS

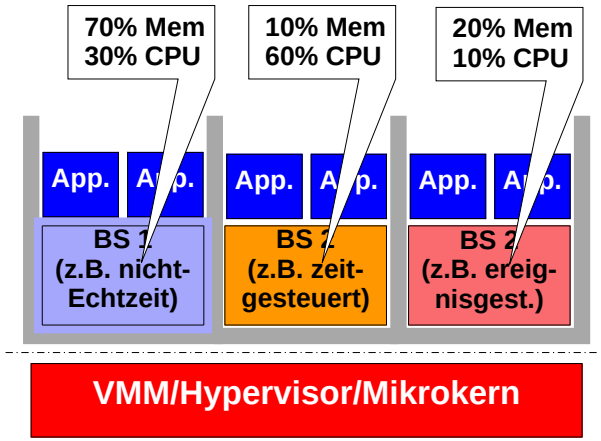
- kleiner Echtzeitkern übernimmt Kontrolle über Interrupt-Hardware
- Nicht-Echtzeitsystem als „idle-Task” des Echtzeitkerns
- Beispiele:
  - Ardence RTX (Windows)
  - RTLinux, RTAI (Linux)



- Vor-/Nachteile:
  - + einfache Möglichkeit zum „Nachrüsten” von Echtzeitfunktionalität
  - + gutes Echtzeitverhalten
  - - keine erzwungene Trennung der Schnittstellen, große „trusted code base”
    $\Rightarrow$ problematisch für sicherheitskritische Anwendungen
  - - keine Schnittstellenwahl, EZ- und nicht-EZ-System bedingen sich gegenseitig

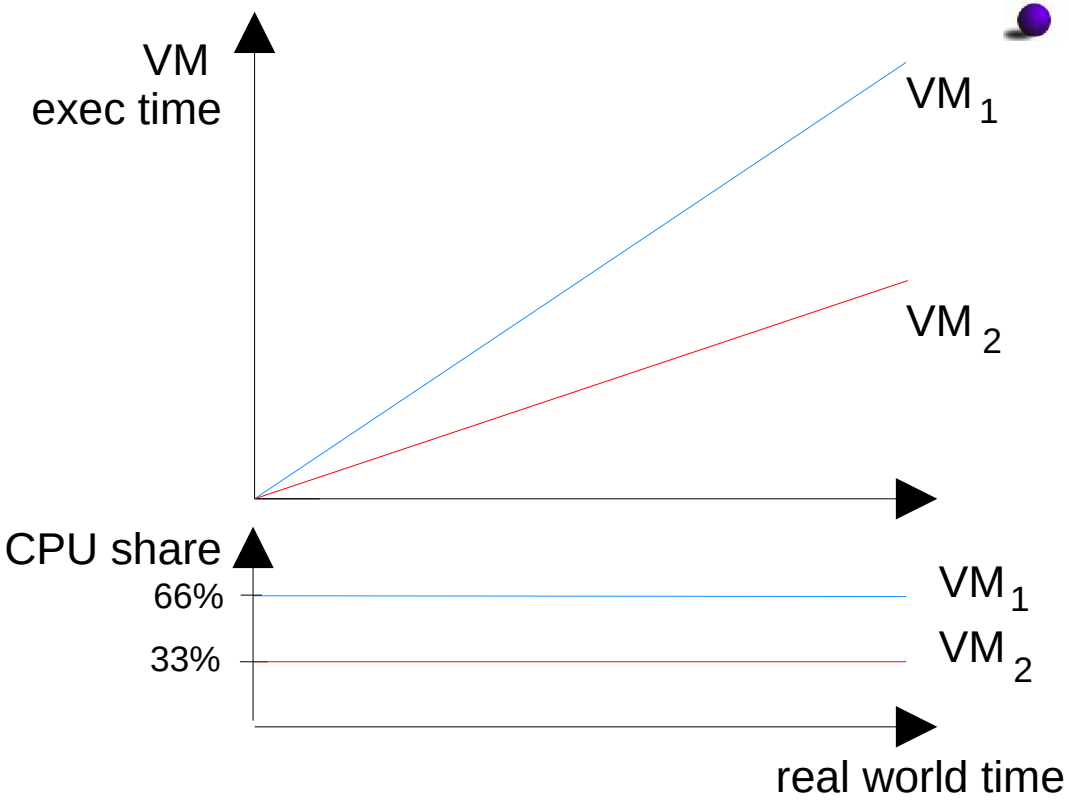
Ansatz 2: Virtualisierung

- Hypervisor/Mikrokern/
  Virtual Machine Monitor verteilt die Betriebsmittel
- jede VM enthält ein eigenes Betriebssystem
- Beispiele:
  - IBM System/360 (historisch)
  - VMware, Xen (aktuell)



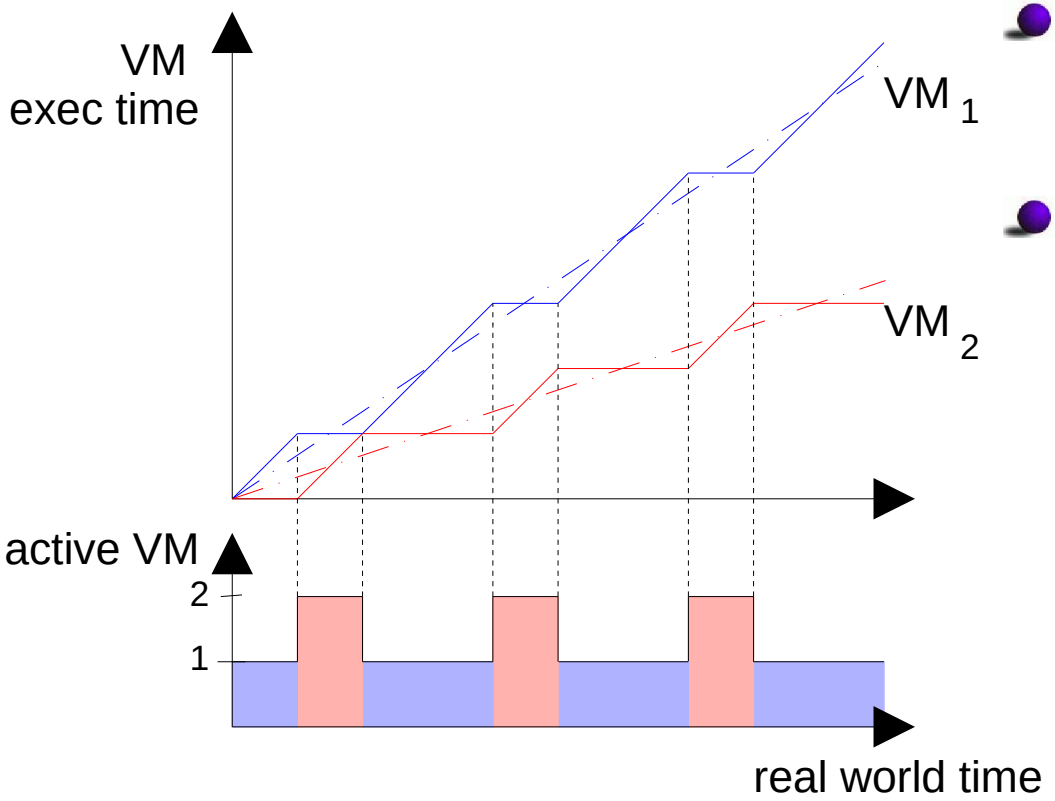
- Vor-/Nachteile:
  - + streng getrennte Betriebsmittel, VMs sind vollkommen unabhängig
  - + kleine „trusted code base” (i.W. der Mikrokern)
    $\Rightarrow$ auch für komplexe sicherheitskritische Anwendungen
  - - ursprünglich für Serverkonsolidierung gedacht, nur bedingt echtzeittauglich

Virtualisierung: Zeitverhalten (1)



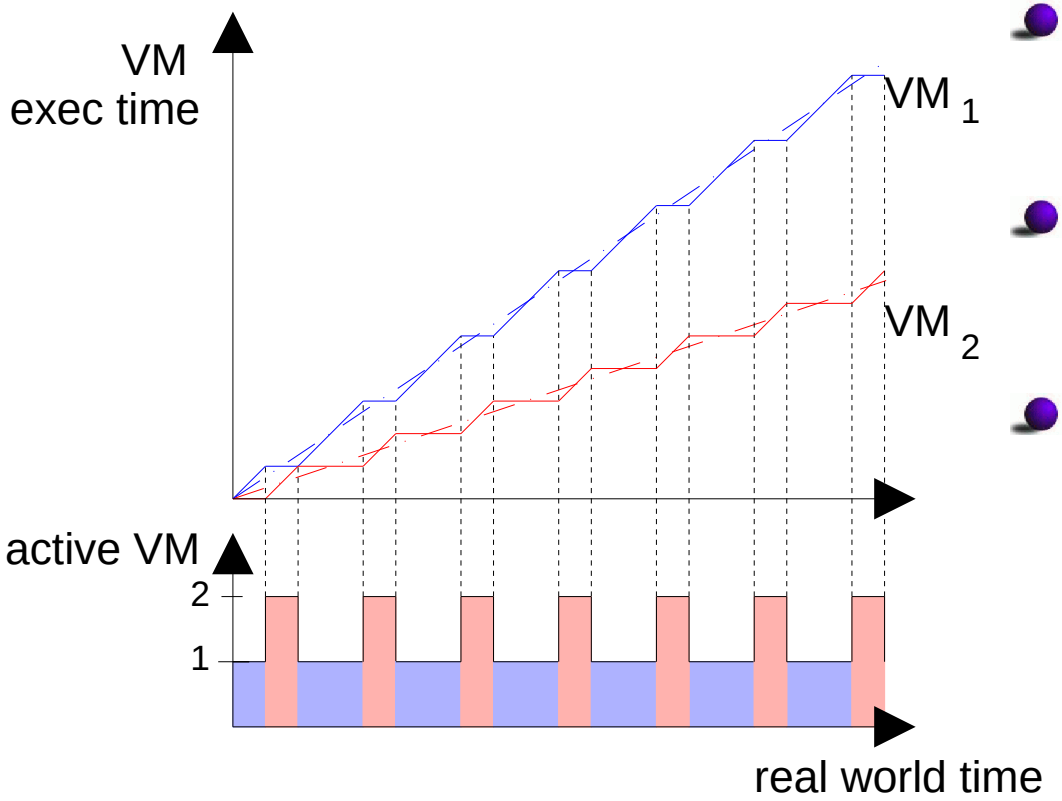
- idealisiert: anteilige Prozessorzuteilung

Virtualisierung: Zeitverhalten (1)



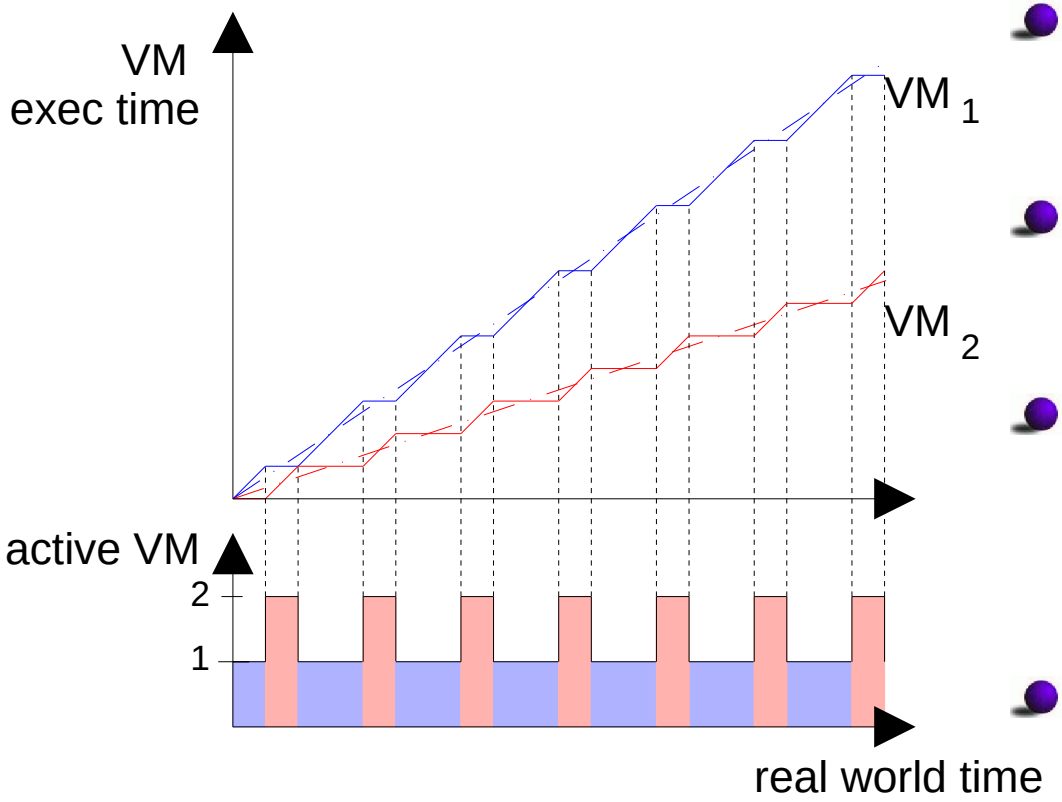
- idealisiert: anteilige Prozessorzuteilung
- real: Approximation per Zeitscheiben-Verfahren

Virtualisierung: Zeitverhalten (1)



- idealisiert: anteilige Prozessorzuteilung
- real: Approximation per Zeitscheiben-Verfahren
- Approximation wird mit kürzer werdender Zeitscheibe präziser

Virtualisierung: Zeitverhalten (1)



- idealisiert: anteilige Prozessorzuteilung
- real: Approximation per Zeitscheiben-Verfahren
- Approximation wird mit kürzer werdender Zeitscheibe präziser
- Begrenzung gegeben durch Umschaltverluste

Virtualisierung: Zeitverhalten (2)

SYSGO REAL-TIME SOLUTIONS

- Annahme anteiliger Prozessorzuteilung:
  - deterministisches Zeitverhalten grundsätzlich möglich
  - erzielbare Latenzzeiten, Jitter, etc. liegen jedoch unter realistischen Annahmen um 1-2 Größenordnungen über denen eines ohne Virtualisierung arbeitenden Systems
  - Verbesserung wäre möglich durch Synchronisation des Gastsystems
- aber: bei Xen und VMware
  - VM-Zeitscheibendauern variieren (abhängig vom Verhalten der Gastsysteme)
  - $\Rightarrow$ keine Vorhersagen über Zeitverhalten eines Gastsystems möglich

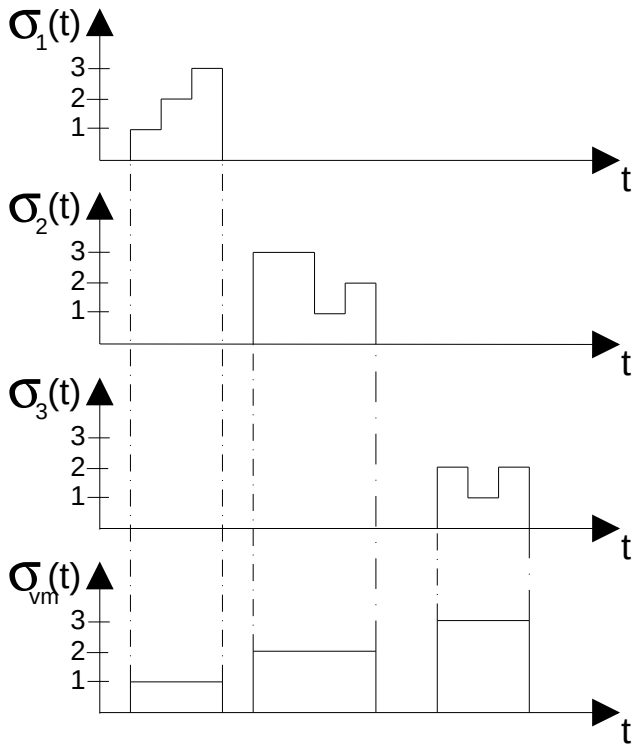
Virtualisierung für EZ-Systeme

SYSGO REAL-TIME SOLUTIONS

- PikeOS: spezialisierte Virtualisierungsumgebung für Echtzeitanwendungen
- „para”-Virtualisierung (ähnlich Xen)
- Unterschiedliche Planungsverfahren für unterschiedliche Klassen von Gastsystemen:
  1. Echtzeit, zeitgetrieben
  2. Echtzeit, ereignisgetrieben
  3. Nicht-Echtzeit

Anforderungen(1): Determiniertheit

zeitgetriebene Gastsysteme:



- VM-Ausführungsplan wird als „umschließender" Ausführungsplan aller zeitgetriebenen Gastsysteme definiert.
- setzt voraus, dass die Ausführungspläne der Gastsysteme ...
  - .. sich nicht überschneiden
  - .. die gleiche Periodendauer besitzen

„umschließender" Ausführungsplan ist Funktion der Zeit

Anforderungen (2): Reaktion

SYSGO REAL-TIME SOLUTIONS

ereignisgetriebene Gastsysteme:

- Möglichkeit: Zeitscheibe für Bearbeitung anstehender Ereignisse, zu langsam
- notwendig: unterbrechbare VMs
- steht im Widerspruch zur vorherigen Anforderung (Determiniertheit)
- Sicherheitsrisiko
- $\Rightarrow$ darf nur für vertrauenswürdige Programme erlaubt sein
- keine generelle Lösung möglich $\rightarrow$ System muss ausreichende Flexibilität bieten

Anforderungen (3): Re-Allokation

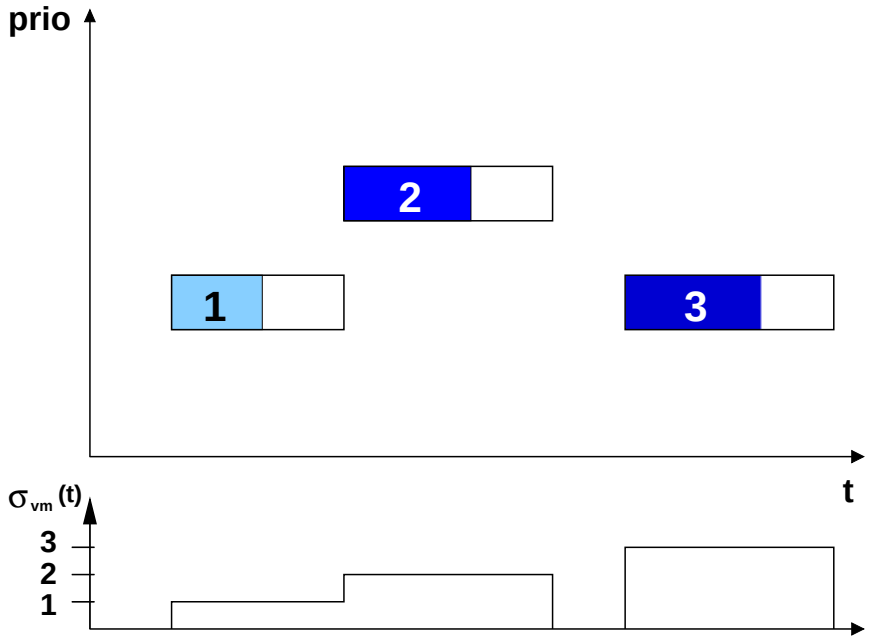
SYSGO REAL-TIME SOLUTIONS

Nicht-Echtzeit VMs:

- Zeitzuteilung für Echtzeit-VMs dimensioniert unter „worst-case”-Annahmen
- Echtzeit-VMs benötigen äusserst selten ihre gesamte Zeit
- ungenutzte Rechenzeit dynamisch für nicht-Echtzeit VMs umwidmen
- **Zudem:** „verhungern” von VMs muss auszuschließen sein
- bei mehreren nicht-Echtzeit VMs muss die zugeteilte Zeit gleich verteilt sein

Vorgehensweise (1)

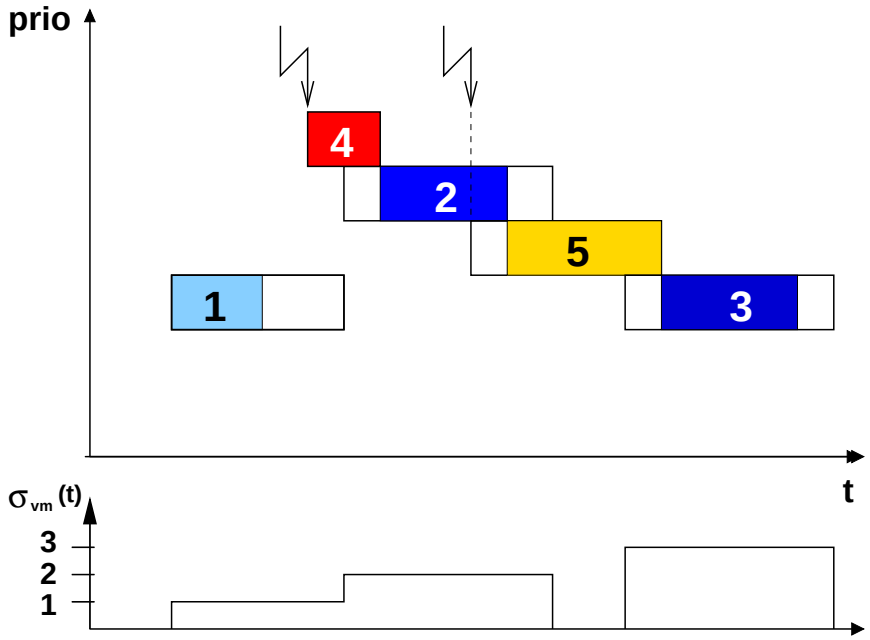
Grundidee: Kombination von zeit- und prioritätsgesteuerter Planung:



- Streng zeitgesteuerter Ausführungsplan für zeitgetriebene VMs.

Vorgehensweise (1)

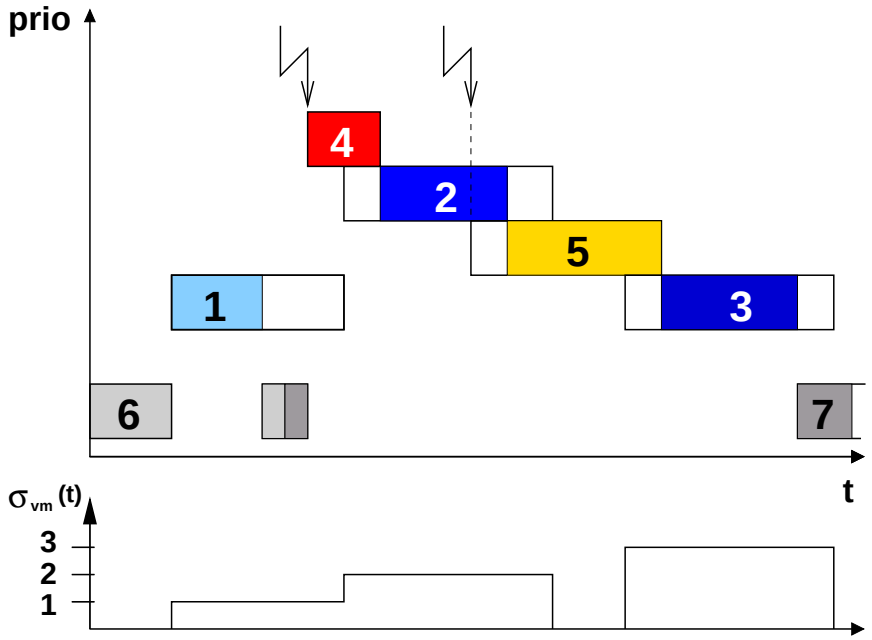
Grundidee: Kombination von zeit- und prioritätsgesteuerter Planung:



- Streng zeitgesteuerter Ausführungsplan für zeitgetriebene VMs.
- Andere VMs konkurrieren anhand ihrer Priorität:
  - Höher ⇒ kann zeitgetriebene VMs unterbrechen
  - Niedriger ⇒ erhält von höherprioren VMs nicht genutzte Zeit

Vorgehensweise (1)

Grundidee: Kombination von zeit- und prioritätsgesteuerter Planung:



- Streng zeitgesteuerter Ausführungsplan für zeitgetriebene VMs.
- Andere VMs konkurrieren anhand ihrer Priorität:
  - Höher ⇒ kann zeitgetriebene VMs unterbrechen
  - Niedriger ⇒ erhält von höherprioren VMs nicht genutzte Zeit
  - Gleich ⇒ gleichmäßige Aufteilung der Zeit

Vorgehensweise (2)

SYSGO REAL-TIME SOLUTIONS

Erste Implementierung: PikeOS Mikrokern

- VMs werden als Gruppen von Prozessen repräsentiert:

  synchron: "VM container"

  asynchron: Event-/Interrupt-Handler

- Prozesse erhalten Prioritäten und *Zeitdomänen*.
- Prozesse arbeiten nur, wenn ihre Zeitdomäne aktiv ist (unabhängig von Priorität).
- Zeitdomänen werden als Feld von (FIFO) Bereit-Listen mit festen Prioritäten repräsentiert.
- $\Rightarrow$ Klassische, prioritätengesteuerte FIFO-Prozessplanung in jeder Zeitdomäne

Vorgehensweise (3)

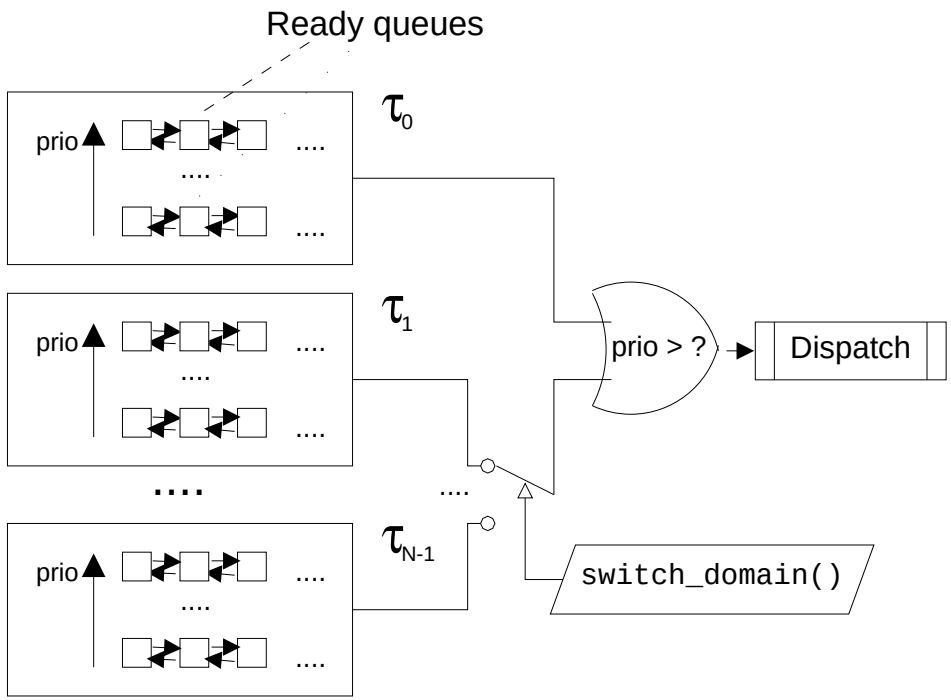
SYSGO REAL-TIME SOLUTIONS

- Ähnlich wie bei ARINC 653 (”partition scheduling”) werden Zeitdomänen zyklisch aktiviert
- Anders als bei ARINC 653 können zwei gleichzeitig aktiv sein:

  τ_0: *Hintergrund* domäne: stets aktiv

  τ_i: *Vordergrund* domäne: eine von $N-1$ Zeitdomänen, zyklisch umgeschaltet
- Prozesse aus τ_0 und der gerade aktiven τ_i konkurrieren anhand ihrer Priorität.

Vorgehensweise (4)



- Mikrokern implementiert lediglich den Mechanismus zur Umschaltung
- Umschalt-Strategie wird durch (vertrauenswürdiges) Anwenderprogramm bestimmt
  $\Rightarrow$ Möglichkeit, beliebige Strategien umzusetzen

Praktische Umsetzung

SYSGO REAL-TIME SOLUTIONS

PikeOS Mikrokern

- Konzeptionell basiert auf „L4” (Liedtke 1995)
- Derzeit: lauffähig auf PowerPC, ia-32 und MIPS
- Verfügbare Gastsysteme: Linux, POSIX threads (PSE51), ARINC Apex, Ada Runtime, JVM, OSEK OS, ...
- Zeitscheibendauern bis herab zu 500 μs möglich (PowerPC MPC5200@400 MHz, 10% Umschaltverlust)

Nächste Schritte

SYSGO
REAL-TIME SOLUTIONS

- Multiprozessor- (Multicore-) Unterstützung
  - Binden von zeit- bzw. ereignisgetriebenen Gastsystemen an verschiedene Prozessoren
  - „coscheduling” paralleler Echtzeitapplikationen
- verwende Xen als Testbett

Zusammenfassung

SYSGO
REAL-TIME SOLUTIONS

- Vielfalt an Zeitanforderungen alltäglicher Echtzeitsysteme erfordert vielfältige Betriebssystemschnittstellen
- mögliche Basistechnologien: „Kern-im-Kern” Systeme, Virtualisierung
- Virtualisierung ist überlegen, aber bekannte Implementierungen sind nur bedingt echtzeittauglich
- vorgestellte Vorgehensweise ermöglicht Koexistenz von VMs mit unterschiedlichen Zeitanforderungen.
- Koexistenz von zeit- und ereignisgetriebenen Systemen bleibt problematisch
- Die vorgestellte Vorgehensweise bietet hier (zumindest) große Flexibilität.

SYSGO
REAL-TIME SOLUTIONS

Danke für Ihre Aufmerksamkeit!