Zustellerkonzepte in Mehrkern-Systemen

Moritz König

12. Januar 2016

Gliederung

Motivation

- Bisherige Vorträge: Einplanung von periodischen Aufgaben
- Aber: nicht periodische Aufgaben auch wichtig
  z.B. jederzeit mögliche Nutzereingabe, Fehlerbehandlung

  $\Rightarrow$ neues Ziel:
- möglichst gutes Antwortverhalten
- ohne Fristverletzung von periodischen Aufgaben

  Probleme dabei sind:
- wenig Wissen über Ankunftszeit der nicht periodischen Aufgaben
  $\Rightarrow$ dynamische Einplanung nötig
  $\Rightarrow$ erschwert die Kombination der obigen Teilziele

Grundsätzliche Herangehensweisen

Unterbrecherbetrieb

- Prinzip:
  - Verdrängung aktuell ausgeführter periodischer Aufgaben
  - sofortige Ausführung ausgelöster nicht periodischer Aufgaben
- Vor- und Nachteile:
  - minimierte Antwortzeiten
  - anfällig für Überlast
  - mögliche Fristverletzung

$\Rightarrow$ für harte Echtzeitanforderungen ungeeignet

Hintergrundbetrieb

- Prinzip:
  - Fertigstellung periodischer Aufgaben
  - Ausführung nicht periodischer Aufgaben danach
- Vor- und Nachteile:
  - garantierte Termineinhaltung
  - sehr schlechtes Antwortverhalten

$\Rightarrow$ wird der Zielsetzung ebenfalls nicht gerecht

Slack-Stealing

- Prinzip:
  - Aufschieben periodischer Aufgaben ohne Fristverletzung
  - Ausführung der nicht periodischen Aufgaben in gewonnener Zeit (Schlupf)
- Vor- und Nachteile:
  - guter Kompromiss:
    - gutes Antwortverhalten
    - garantierte Fristeinhaltung
  - Für ereignisgesteuerte Systeme jedoch zu kompliziert (Berechnung der Schlupfzeit)

Periodischer Zusteller

Ein Zusteller hat:

- ein Ausführungsbudget (entspricht der WCET einer periodischen Aufgabe)
- eine Auffüllperiode (entspricht der Periode)

Mögliche Implementierungen:

- abfragend:
  Überprüfung auf nicht periodische Aufgaben bei Einlastung
  - falls vorhanden: Nutzung des Budgets zur Ausführung
  - sonst: Verfallen des Budgets

  Nachteil bei sporadischen Aufgaben: hohe Abfragerate erforderlich,
  Bei aperiodischen vielleicht wünschenswert
  $\Rightarrow$ großer Overhead durch Abfragen
  $\Rightarrow$ schlechtes Antwortverhalten oder Überlast

Bandbreite bewahrende Zusteller

Bandbreite bewahrende Zusteller:

- aufschiebbar:
  - bewahrt Budget bei Untätigkeit
  - verbraucht Budget bei Ausführung mit einer Rate von $\frac{1}{Zeiteinheit}$
  - erhält nach jeder Auffüllperiode das volle Budget
  - verhält sich nicht wie periodische Aufgabe
- sporadisch:
  - verschiedene Regeln zum Verbrauch und Auffüllung des Budgets möglich
  - verhält sich wie periodische Aufgabe
  - öfter planbar als aufschiebbarer Zusteller

Verfahren zur Verbesserung des Antwortverhaltens

Probleme des Zustellerbetriebs

- von der Einplanung abhängige Probleme:
  - partitioniert: geringere Auslastung, Mehrfachausführung verhindern
  - global: Synchronisation nötig
- großes Problem: Vereinung der folgenden Ziele:
  - garantierte Echtzeitfähigkeit
  - maximale Auslastung der Prozessoren

  Bsp.: EDF-Einplanung optimal, Auslastung ist jedoch strikt unter der maximalen
- Ziel: Aufteilung der freien Kapazität auf aperiodische Aufgaben
  - besonders problematisch bei Mehrkern-Systemen:
    Einzelne Arbeitsaufträge unzerteilbar (bzw. nicht sinnvoll oft zerteilbar)
    $\Rightarrow$ kleine Fragmente schwierig nutzbar
  - Antwortzeitoptimierung schwierig
- Aber: in kritischen Situationen Auslastung unter Maximum!

Verfahren zur Verbesserung des Antwortverhaltens

- Idee:
  Behandlung der nicht periodischen Aufgaben:
  - Zuweisung einer Frist
  - Einplanung wie periodische Aufgaben
- Echtzeitgarantie
  - bei weniger als 50% Auslastung für globale Einplanung
  - bei weniger als 31% Auslastung für partitionierte Einplanung

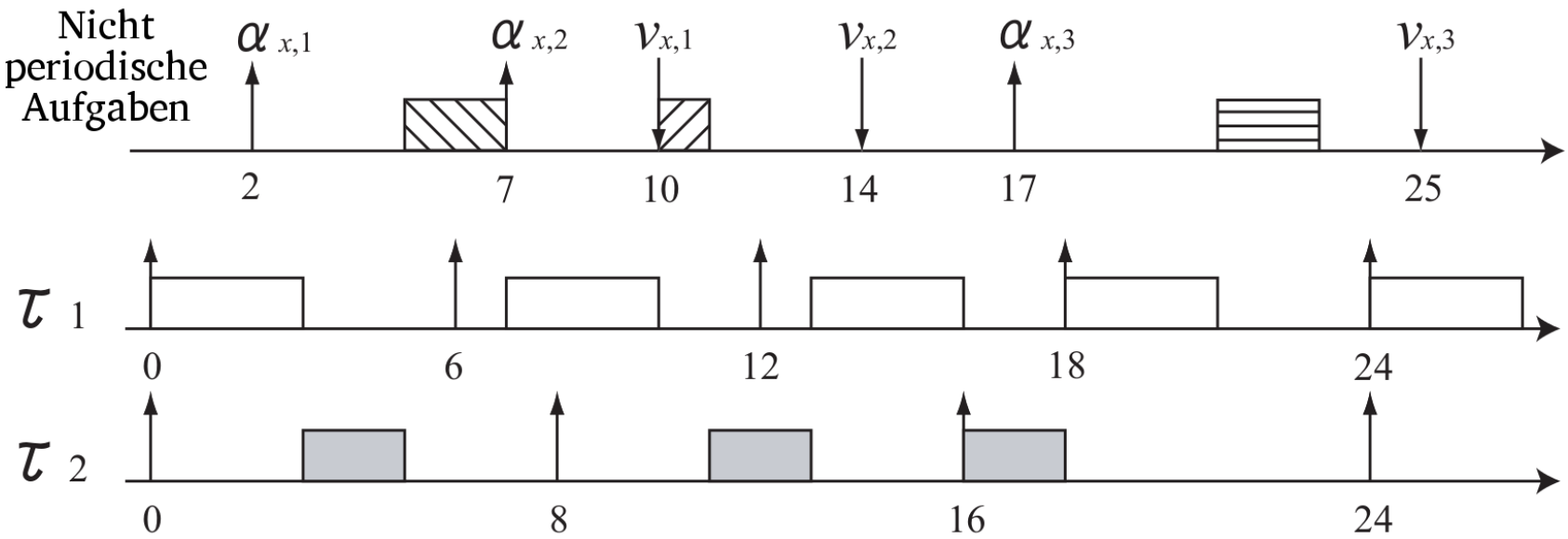
Funktionsweise des TBS-Algorithmus

- Einplanung über EDF
- Wahl einer möglichst frühen virtuellen Frist:
  Für die k-te nicht periodische Aufgabe $\alpha_{x,k}$ auf Prozessor x ergibt sich eine Frist $v_{x,k}$:

$$v_{x,k} = max\{Ankunftszeit\ von\ \alpha_{x,k}, v_{x,k-1}\} + \frac{WCET\ von\ \alpha_{x,k}}{Bandbreite\ des\ Zustellers}$$

  mit $v_{x,0} := 0$

Beispiel einer Einplanung mittels TBS-Algorithmus

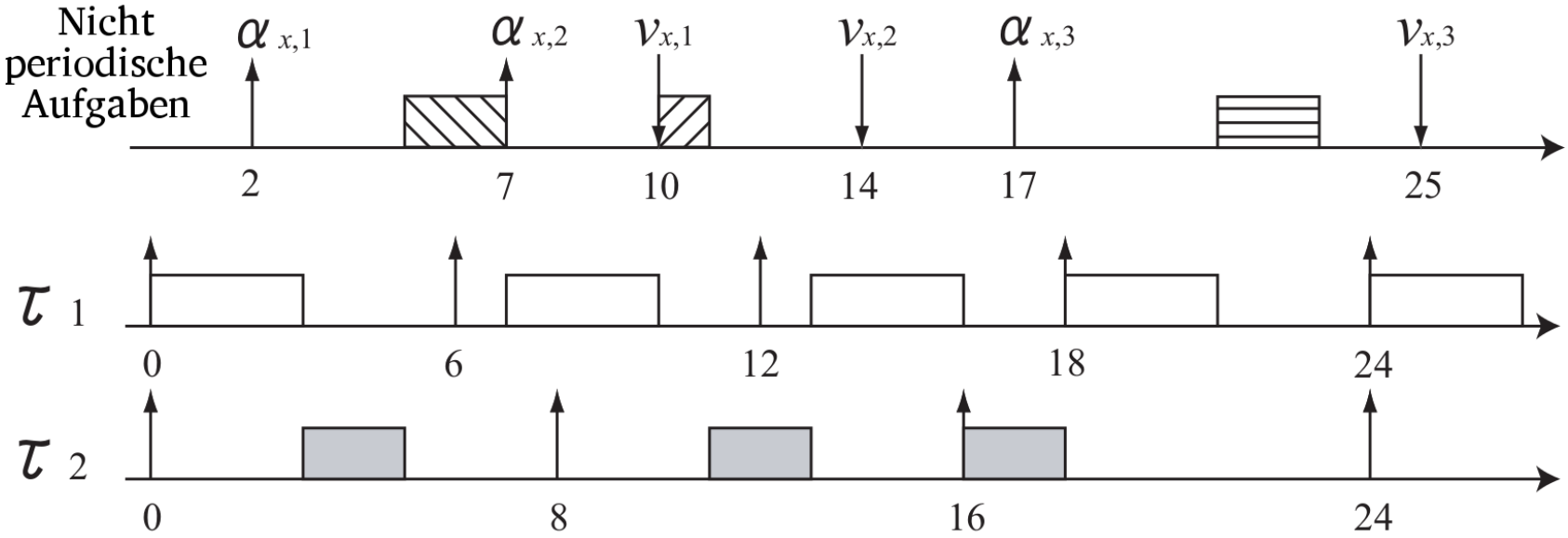


2 periodische Aufgaben:

- τ_1 mit Periode 6 und WCET 3
- τ_2 mit Periode 8 und WCET 2

$\Rightarrow$ Bandbreite des Zustellers = 1 - Auslastung der periodischen Aufgaben = 0.25

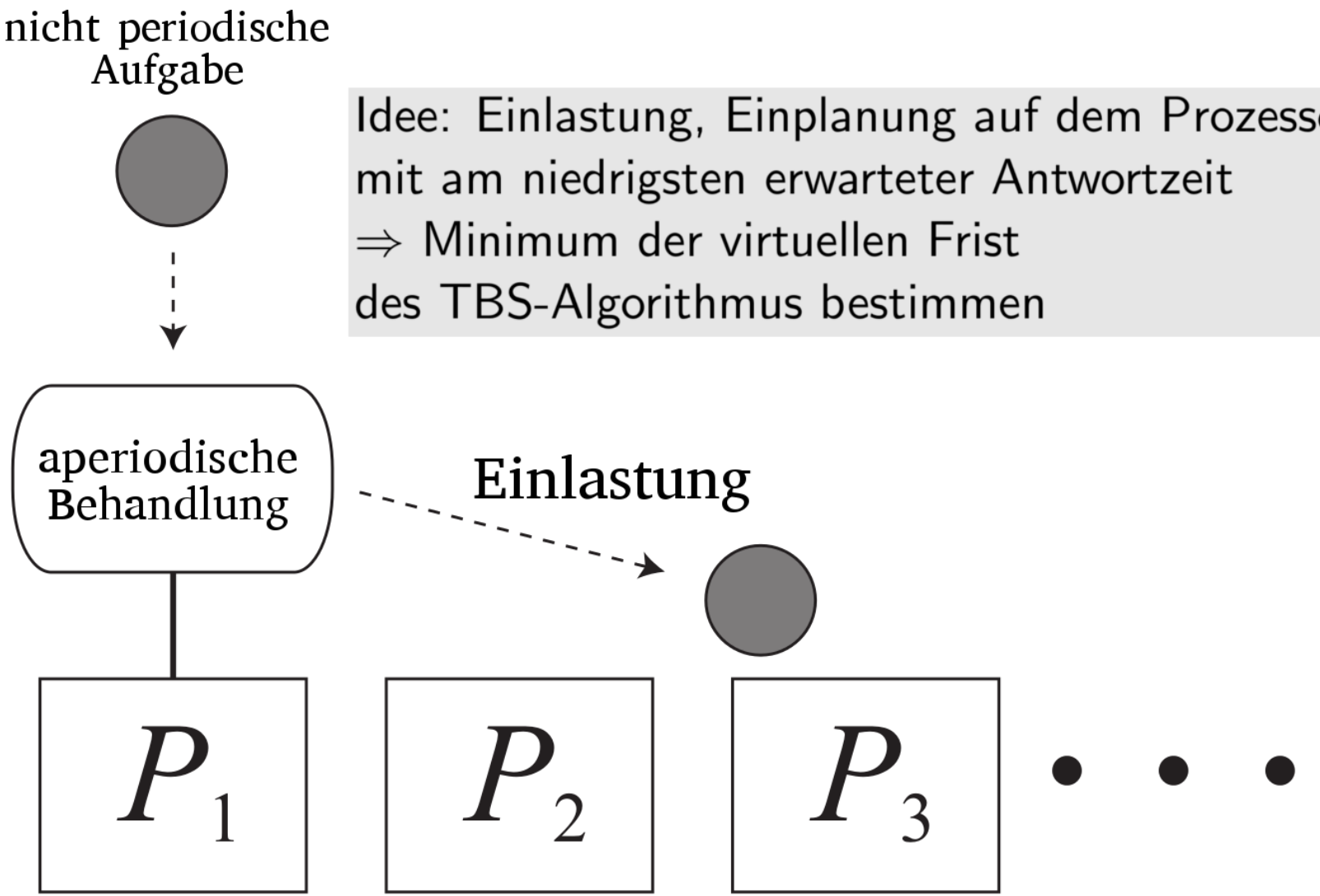
Beispiel einer Einplanung mittels TBS-Algorithmus



virtuelle Fristen der nicht periodischen Aufgaben $\alpha_{x,k}$:

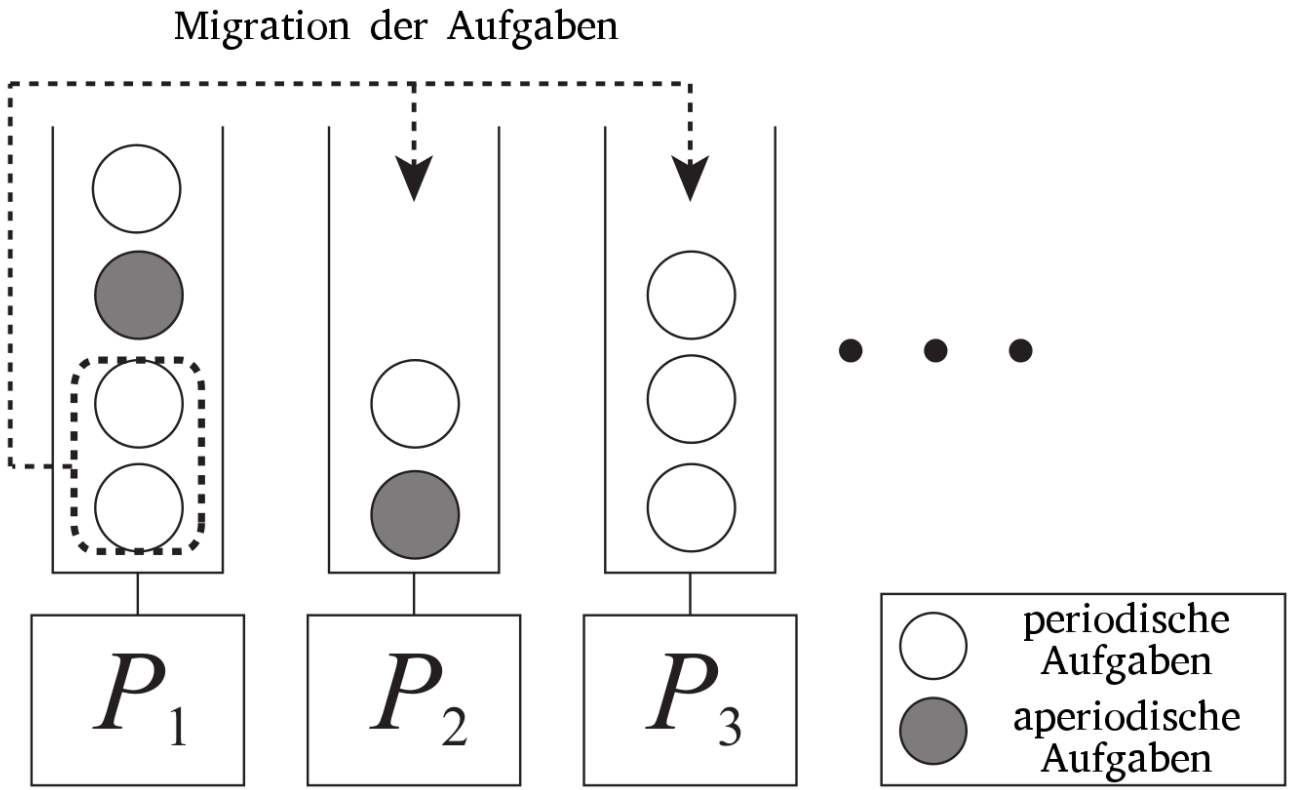
- $v_{x,1} = 2 + \frac{2}{0.25} = 10$
- $v_{x,2} = 10 + \frac{1}{0.25} = 14$
- $v_{x,3} = 17 + \frac{2}{0.25} = 25$

Funktionsweise des Einlastungsverfahrens



Funktionsweise des Migrationsverfahrens

Idee: Migration höher priorisierter periodischer Aufgaben auf andere Prozessoren, falls möglich



Algorithmus des Migrationsverfahrens

1.

- Auswahl des periodischen Auftrags mit frühester Frist
- Wahl des nächstbesten bei Mehrfachmigration
- Abbruch bei keinem Fund

2.

- Auswahl eines Prozessors ohne Fristverletzung

3.

- Migration des Arbeitsauftrags
- Behandlung als aperiodische Aufgabe mit Frist:
  $max\{Migrationszeit, Frist\ der\ vorherigen\ Aufgabe\} + \frac{Restausführungszeit}{Zustellerbandbreite}$

4.

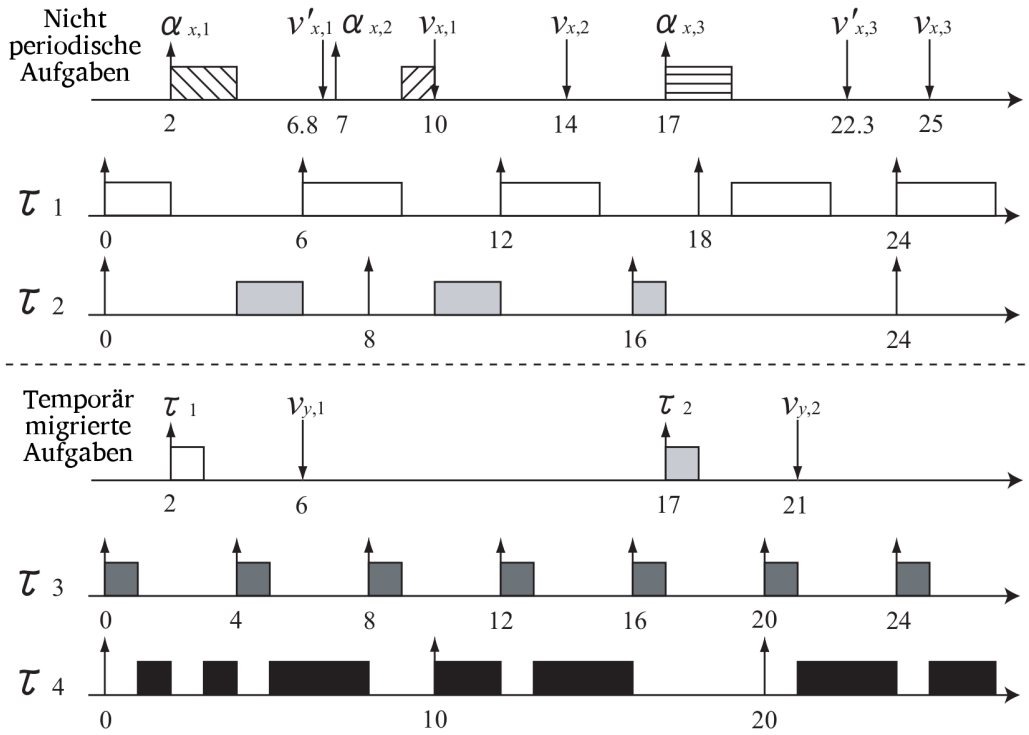
- Verbesserung der virtuellen Frist der tatsächlich aperiodischen Aufgabe auf: $max\{Migrationszeit, Frist\ vorherige\ Aufgabe\} + \frac{WCET}{Zustellerbandbreite + \frac{Restausführungszeit\ des\ migrierten\ Arbeitsauftrags}{Periode\ des\ migrierten\ Arbeitsauftrags}}$

5.

- Rückmigration der periodischen Aufgabe für nächsten Arbeitsauftrag

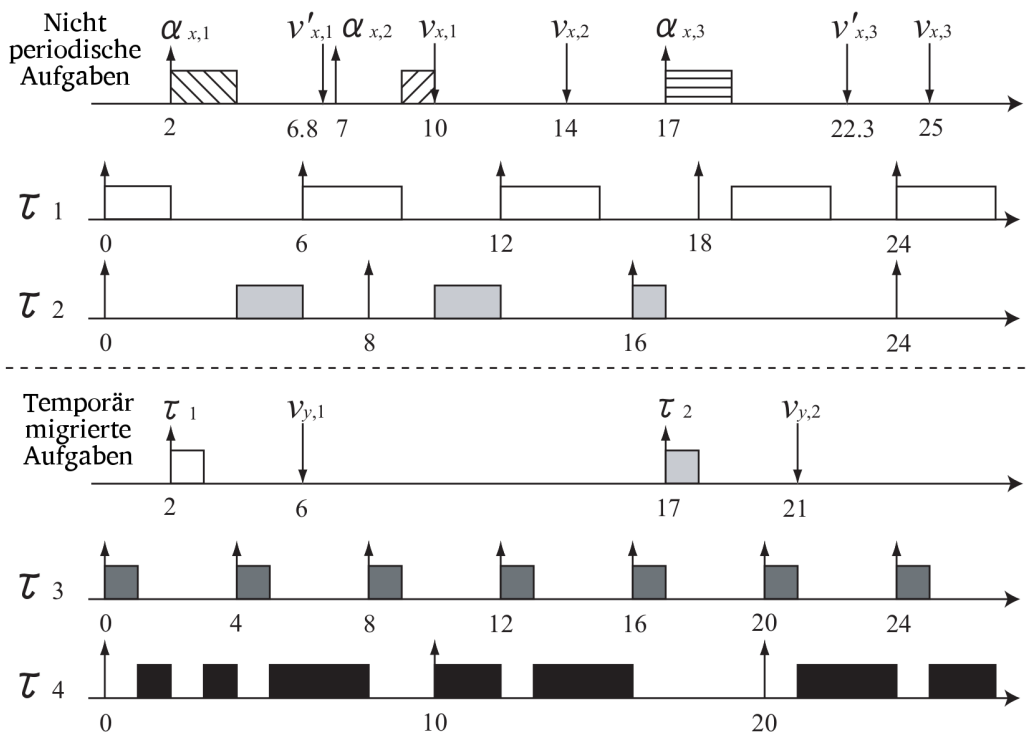
Anmerkung: vorherige Aufgabe bezieht sich immer auf den Prozessor, auf dem die Aufgabe eingeplant wird

Einplanungsbeispiel mittels Migrationsverfahren



- Nicht periodische und periodische Aufgaben auf Prozessor P_x wie im TBS-Beispiel
- keine nicht periodischen Aufgaben auf P_y, 2 periodische:
  - τ_3 mit Periode 4 und WCET 1
  - τ_4 mit Periode 10 und WCET 5 $\Longrightarrow$ selbe Zustellerbandbreite

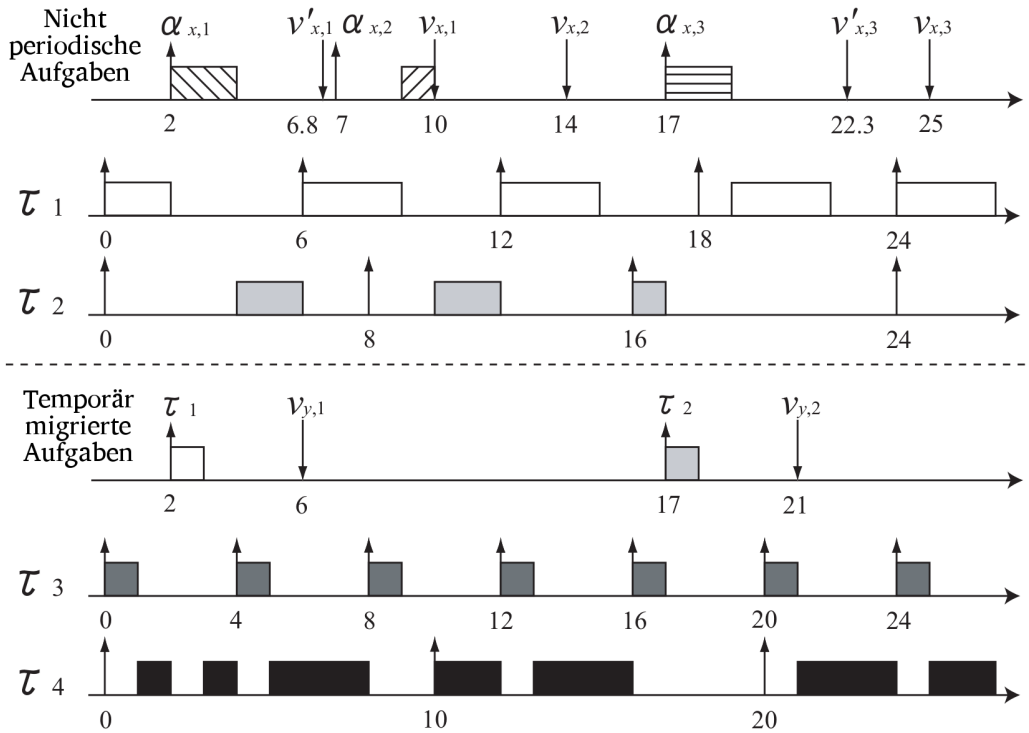
Einplanungsbeispiel mittels Migrationsverfahren



Einplanung von $\alpha_{x,1}$:

- Migration von $\tau_{1,1}$ auf P_y möglich $\Rightarrow$ virtuelle Frist: $2 + \frac{1}{0.25} = 6$
- virtuelle Frist von $\alpha_{x,1}$ verbessert sich von 10 auf $2 + \frac{2}{0.25+\frac{1}{6}} = 6.8$

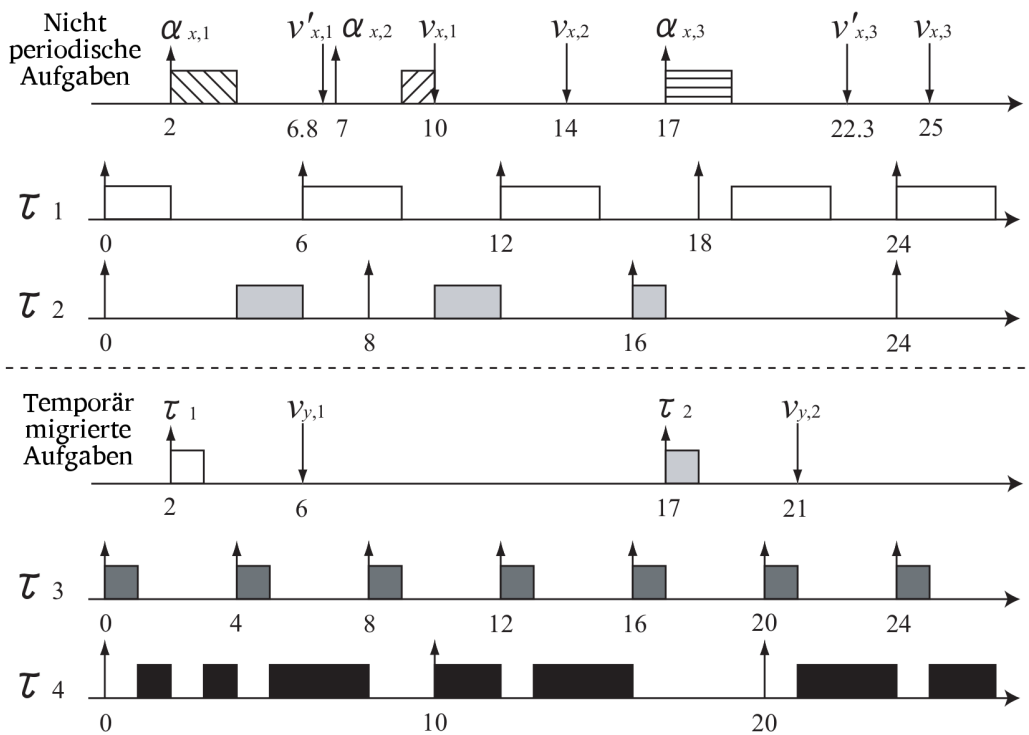
Einplanungsbeispiel mittels Migrationsverfahren



Einplanung von $\alpha_{x,2}$:

- Migration von τ_1 auf P_y nicht möglich, da Fristverletzung
- Also keine verbesserte Frist für $\alpha_{x,2}$

Einplanungsbeispiel mittels Migrationsverfahren



Einplanung von $\alpha_{x,3}$:

- Migration von τ_2 auf P_y möglich $\Rightarrow$ virtuelle Frist: $17 + \frac{1}{0.25} = 21$
- virtuelle Frist von $\alpha_{x,3}$ verbessert sich von 25 auf $17 + \frac{2}{0.25+\frac{1}{8}} = 22.3$

Auswirkungen des Migrationsverfahrens

- auf das Antwortverhalten:
  - temporäre Abwesenheit von periodischen Aufgaben
    $\Rightarrow$ größere Zusteller-Bandbreite $\Longrightarrow$ frühere Fristen möglich
  - weniger periodische Aufgaben auf betrachtetem Prozessor
    $\Rightarrow$ frühere Einlastung auch ohne frühere Fristen
  - bereits angekommene aperiodische Aufgaben nicht beeinträchtigt
  - Einlastung später ankommender Aufgaben erst nach Vollendung der migrierten Aufgabe(n)
  - Gesamt:
    - Verbesserung des Antwortverhaltens
    - weniger effektiv als das Einlastungsverfahren
- auf die Planbarkeit:

planbar ohne Optimierung $\Longrightarrow$ auch mit Verfahren planbar

Zusammenfassung

- Vorgehensweisen bei der Einplanung von nicht periodischen Aufgaben:
  - Unterbrecherbetrieb:
    gutes Antwortverhalten, keine Echtzeitgarantie
  - Hintergrundbetrieb:
    Einhaltung von Fristen, schlechtes Antwortverhalten
  - Slack-Stealing:
    guter Kompromiss
    ABER: auf vorranggesteuerten Systemen nicht praktikabel
  - periodischer Zusteller
- Verbesserung des Antwortverhaltens:
  - TBS:
    frühe virtuelle Frist, Einplanung über EDF
  - Einlastungsverfahren für beliebig ausführbare Aufgaben:
    Wahl der frühest möglichen Frist unter den Prozessoren
  - Migrationsverfahren für Aufgaben, die lokal ausgeführt werden müssen:
    Migration höher priorisierter periodischer Arbeitsaufträge

Danke für eure Aufmerksamkeit.

Fragen?