5 Prozessorvergabe in Multiprozessorsystemen

5.1 Aufgabenstellung

❑ **Vom Betriebssystem zu unterstützende Ziele**

- **Hoher Wirkungsgrad (performance)**
- **Ausbaufähigkeit (scalability)**
- **Hohe Verfügbarkeit (availability) und Zuverlässigkeit (reliability), sanfter Leistungsabfall bei Teilausfällen (graceful degradation)**

❑ **Spezielle Probleme, die aus dem Verlangen nach hohem Wirkungsgrad und Ausbaufähigkeit resultieren:**

- **Zugriffsschutz in sehr großen Adreßräumen (64 Bit für Adressen)**
- **Vermeidung von Verklemmungen (deadlock prevention)**
- **Ausnahmebehandlung bei sehr vielen Prozessoren**
- **Effizient bearbeitbare Darstellung von asynchron arbeitenden oder bearbeitbaren Einheiten**
- **Bereitstellung angepaßter Interaktionsmechanismen**
- **Betriebsmittelverwaltung (Prozessorzuordnung, Speicherverwaltung)**
- **Parallelisierung des Betriebssystems selbst**

❑ **Detaillierte Betrachtung der Prozessorvergabe**

- **Unix-Prozeß bestehend aus Adreßraum und Aktivitätsträger (heavyweight processes)**
  - Parallelisierung einer Aufgabe nur durch Benutzung mehrerer Prozesse; disjunkte Adreßräume unnatürlich bei Datenpartitionierung
  - Erzeugung, Zuordnung und Tilgung von Prozessen aufwendig; dementsprechend zeitraubend ist der Wechsel zwischen seriellen und parallelen Phasen
  - Prozeßwechsel sind aufwendig wegen des damit verbundenen Kontextwechsels und den daraus resultierenden Cache- und TLB-Invalidierungen

❑ **Detaillierte Betrachtung der Prozessorvergabe (Forts.)**

- **Entkopplung von Adreßraum und Aktivitätsträger durch Kern-Fäden (middleweight processes, kernel-level processes, kernel-threads)**
  - Prozessorvergabe muß nicht mit Kontextwechsel verbunden sein; bei ‘single task’-Betrieb keine Kontextwechsel
  - Bieten eine allgemeine Benutzerschnittstelle
Beispiel: Mach, Windows NT
  - Betriebssystem macht Zuordnung und kann dabei seine Kenntnis des Systemzustandes nutzen, um gute Betriebsmittelauslastung zu erzielen
  - Für feingranulare Parallelität immer noch zu schwerfällig (z. B. ist bei manchen Rechensystemen Sicherung und Wiedereinsetzung der Register für Gleitkommabearbeitung aufwendig)
  - Verwaltungsoperationen für Fäden (z. B. im Rahmen von Interaktionen) erfordern Systemaufruf
  - Es ist unwahrscheinlich, daß eine (auch parametrierte) Zuordnungskonzeption für alle Anwendungen effizient ist.

Prozessorvergabe - Multiprozessoren: Aufgabenstellung

❑ **Detaillierte Betrachtung der Prozessorvergabe (Forts.)**

- **Implementierung von leichtgewichtigen Fäden (lightweight processes, user level threads) durch Multiplexen schwer- oder mittelgewichtiger Fäden**
- **Koordinierung**
  - Gegenseitiger Ausschluß
  - Bedingungsvariable
  - Leser/Schreiber-Koordinierung (optimiert für häufiges Lesen und seltenes Schreiben)
- **Realisierung und Verwaltung durch Bibliotheken**

BP 2

Prozessorvergabe - Multiprozessoren: Aufgabenstellung

◆ **Beispiel: POSIX-Standard**

- **Funktionalität**
  - **Erzeugung von Fäden**
  - **Beendigung von Fäden**
  - **Verwaltung fadenspezifischer Daten**
  - **Signale zwischen Fäden**
  - **Identifikation von Fäden**
  - **Einplanung (Scheduling) von Fäden**
  - **Gegenseitiger Ausschluß zwischen Fäden**
  - **Bedingungsvariable (Monitore)**
  - **Leser/Schreiber-Koordinierung**
  - **Registrierung von Sonderbehandlungen bei fork()**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

BP 2

Prozessorvergabe - Multiprozessoren: Aufgabenstellung

❑ **Einfaches Beispiel**

```
void mainline ( ... ) {
  char int result;
  thread_t helper;
  int status

  thr_create(0, 0, fetch, &result, 0, &helper);

    // do something else for a while

  thr_join(helper, 0, &status);
    // it's now safe to use result
}

void fetch(int *result) {

    // fetch value from a database

  *result = value;
  thr_exit(0);
}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

BP 2

Prozessorvergabe - Multiprozessoren: Aufgabenstellung

❑ **Realisierung durch Bibliotheken**

- **Erfordert Prozessorvergabe auf wenigstens zwei Ebenen:**
  1. **im Betriebssystem**
  2. **auf Anwendungsebene**

  Nachteile:

  Die Verwaltung von Fäden der Anwendungsebene hat keine Kenntnisse über betriebssysteminterne Ereignisse

  Schedulingentscheidungen des Betriebssystems wirken sich in gleicher Weise auf alle Fäden der Anwendungsebene aus, die durch Multiplexen aus einem Kern-Faden hervorgehen (z. B. Blockierung infolge E/A oder wegen Seitenfehlers).

  Kern hat bei seinen Entscheidungen keinerlei Kenntnis von den Fäden der Anwendungsebene.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Prozessorvergabe - Multiprozessoren: Aufgabenstellung

- **Einführung mehrstufiger (vor allem zweistufiger) Scheduler**
  - Benachrichtigung des Schedulers der Benutzerebene über bestimmte Kernereignisse durch 'upcalls'
  - Benachrichtigung des BS-Kerns über alle Ereignisse der Benutzerebene, die für das BS-Scheduling bedeutsam sind.

Prozessorvergabe - Multiprozessoren: Kern-Fäden

5.2 Prozessorvergabe für Kern-Fäden

◆ **Scheduler-Struktur**

- Struktur der Warteschlangen
- Parallele Bearbeitung von Warteschlangen

◆ **Strategien**

- **Statische oder dynamische Zuordnung von Prozessor und Faden bei Mehrbenutzerbetrieb;**

  Ergebnisse der Untersuchungen von Zahorjan und McCann
  (J. Zahorjan and C. McCann. Processor scheduling in shared memory multiprocessors. In *Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer systems*, pages 214-225, May 1990.):
  - Unabhängig von der Einzel- und Gesamtlast ist dynamisches Scheduling am besten, wenn der Zusatzaufwand für Kontextumschaltungen gering ist.
  - Die Vorteile dynamischer Zuordnung werden um so größer, je häufiger sich der 'Parallelitätsgrad' ändert.
  - Die Vorteile dynamischer Zuordnung werden mit zunehmender Systemlast größer.
  - Bezüglich der mittleren Antwortzeit ist dynamisches Zuordnen fast immer besser.

Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Gemeinsame Bereitschlange, aus der sich freie Prozessoren selbst bedienen:**
  - Vorteil:
    Automatische Lastverteilung
  - Nachteile:
    Keine Rücksichtnahme auf häufig interagierende Fäden
    Keine Rücksichtnahme auf Reihenfolgeforderungen (z. B. beschrieben durch Präzedenzgraphen) und damit keine Gesamtoptimierung bezüglich der Aufträge (jobs)

- **Interaktionsorientiertes Scheduling (meist 'coscheduling' oder 'gang scheduling' genannt)**

  Ousterhout entwickelte drei Methoden:

  Matrix-Scheduling (matrix)

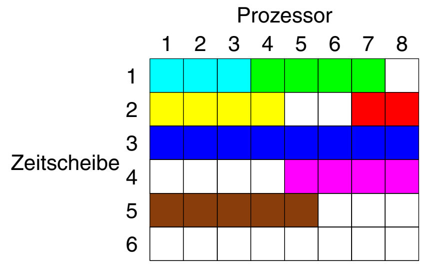
  Fortlaufendes Scheduling (continuous)

  Ungeteiltes Scheduling (undivided)

Prozessorvergabe - Multiprozessoren: Kern-Fäden

◆ **Matrix-Scheduling**

- **Alle Fäden eines Auftrags in einer Zeile einordnen**
- **Fäden einer Zeile werden gleichzeitig zugeordnet**
- **Zeilen nach Round Robin zuordnen**



- **Löcher führen zu Effizienzverlusten**

  ➡ Jeder gemäß ausgewählter Zeile freie Prozessor sucht Faden in seiner Spalte

BP 2

Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Fortlaufend**
  - Matrix linearisiert durch Aneinanderreihung der Zeilen
    ➡ je p aufeinanderfolgende Zellen sind verschiedenen Prozessoren zugeordnet
  - Zu jedem Zeitpunkt Fäden eines Fensters der Länge p zugeordnet
  - Neuankömmling wird in aktives Fenster aufgenommen, wenn dies möglich ist
  - Andernfalls wird Fenster solange nach rechts geschoben, bis zum erstenmal das linke Feld leer ist. Dies wird wiederholt, bis das Fenster genügend freie (nicht notwendigerweise aufeinanderfolgende) Zellen enthält.
    Dadurch Verringerung des Verschnitts!
  - Nach jedem Zeitquant wird Zuordnungsfenster weitergeschoben, bis der linke Eintrag zu einem Auftrag gehört, der im vorherigen Zeitschritt nicht vollständig zugeordnet war.
  - Nachteil: Unzusammenhängend gespeicherte Aufträge können beim Scheduling benachteiligt sein!

BP 2

Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Ungeteilt**
  - Analog fortlaufend, aber nur zusammenhängende Einordnung
  - Der bei fortlaufender Einordnung erwähnte Nachteil verschwindet, dafür größerer Verschnitt
- **Es handelt sich um zentralisierte Algorithmen**

BP 2

Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Anderer Ansatz: baumartig angeordnete Controller verwalten Prozessoren ähnlich einem Buddy-Verfahren bei Speicherplatzvergabe.**
- **Präzedenzgraphorientierte Strategien**
  - **Die meisten Untersuchungen betreffen einstufige Präzedenzgraphen, d. h. ein Hauptfaden erzeugt eine Reihe von Unterfäden und beendet sich dann. Meist wird noch angenommen, daß die Unterfäden nicht interagieren.**
  - **Bekannte Strategie für diesen Fall ist RR**

    Erste Version führt Schlange von bereiten Fäden und bearbeitet sie nach RR, d. h. jeder freigewordene Prozessor bearbeitet den nächsten Auftrag für Q Zeiteinheiten und reiht ihn dann am Ende der Bereitschlange wieder ein.

    Zweite Version führt in der WS Aufträge. Falls ein Auftrag mehr Fäden besitzt als es Prozessoren gibt, wird innerhalb des Auftrags wiederum nach RR zugeteilt.
  - **Probleme:**

    Verdrängung während eines Spinlock oder in kritischem Abschnitt

    Häufige Kontextumschaltungen

BP 2

Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Statische oder dynamische Partitionierung**
  - Ziel ist die Minimierung der Kontextumschaltungen
  - Hypothese: Aufträge erreichen die beste Effizienz, wenn die Zahl der Fäden gleich der Zahl der verwendeten Prozessoren ist.
  - In NUMA-Architekturen existiert häufig eine von der Anwendung vorteilhaft nutzbare Kommunikationsstruktur; wegen ihrer guten Skalierbarkeit spielen Gitterarchitekturen eine besondere Rolle.
  - Als Strategien bieten sich Weiterentwicklungen der Matrixmethode von Ousterhout an.

BP 2 Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Beispiel einer solchen Zuteilungsstrategie für ein torusartiges System, dessen Knoten Multiprozessoren sind (MEMSY):**

Definition

Eine *Zeitscheibenklasse (ZSK)* eines Parallelrechnersystems PS ist ein virtuelles Knotenrechnersystem, das folgende Eigenschaften hat:

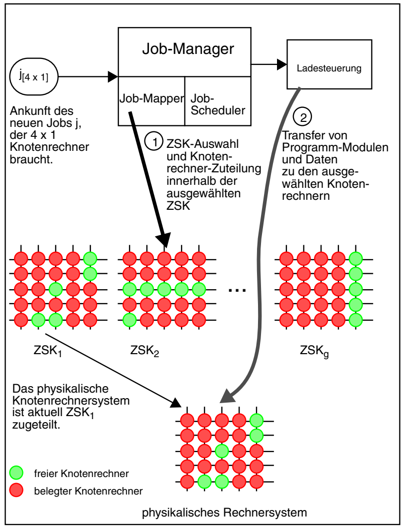
(a) Die Anzahl der virtuellen Knotenrechner in ZSK ist gleich der Anzahl der physikalischen Knotenrechner in PS.

(b) ZSK hat die gleiche Verbindungstopologie wie PS.

(c) Es gibt eine bijektive Abbildung vps_{ZSK} der Knotenrechner von ZSK auf die Knotenrechner von PS. Gilt (für ein $V \in ZSK$ und ein $R \in PS$) $R = vps_{ZSK}(V)$, so sind V und R *topologisch äquivalent*.
Dabei wird unter dem Begriff der *topologischen Äquivalenz* hier folgendes verstanden: Wenn den Systemen PS und ZSK jeweils das gleiche Koordinatensystem zugrundegelegt würde, das die Knotenrechner bijektiv auf Koordinaten abbildet, so wären ein Knotenrechner $K_v \in ZSK$ und ein Knotenrechner $K_r \in PS$ topologisch äquivalent, wenn ihnen die gleichen Koordinaten zugeordnet wären.

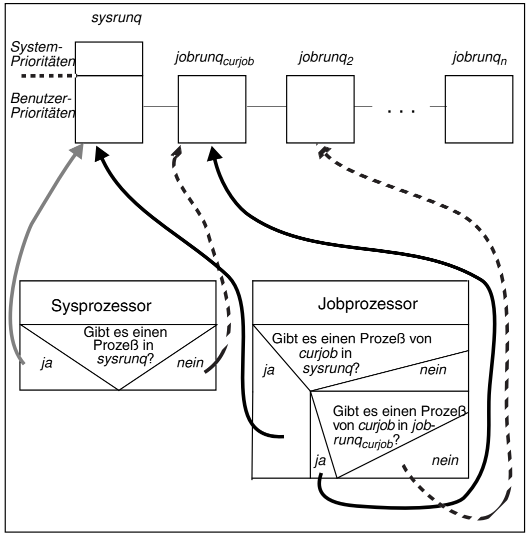
BP 2 Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Einordnung neuer Aufträge**



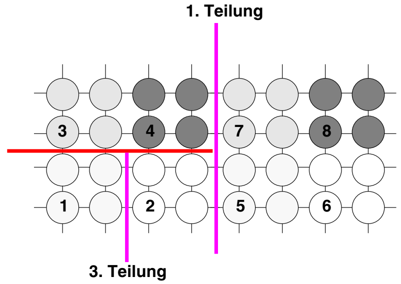
BP 2 Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Knotenscheduling**



BP 2 Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Anpassung der eindimensionalen Buddy-Strategie**



Ein Torus(8 x 4)-System, das aus Torus(2 x 2)-System-Einheiten basierend auf einer zweidimensionalen Buddy-Strategie aufgebaut wird.
Die Zahlen in den Knoten notieren die Reihenfolge, in der die entsprechenden (2 x 2)-Systeme hinzugefügt werden.

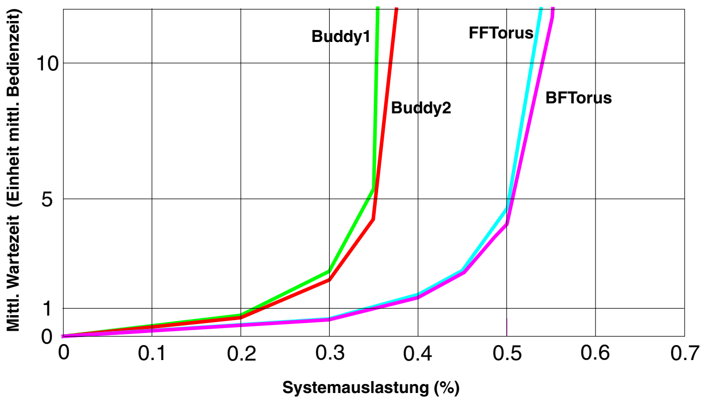
Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Vergleich einiger ausgewählter Strategien mittels Simulation**
  - **FFTorus: First-Fit angepaßt an Oberfläche eines Torus ohne Vorzugsrichtungen bei der Plazierung von Rechtecken**
  - **BFTorus: Best-Fit angepaßt an Oberfläche eines Torus ohne Vorzugsrichtungen bei der Plazierung von Rechtecken**
  - **Buddy1: Teilung gemäß Buddy-Strategie, vollständige Belegung eines 'Buddy'**
  - **Buddy2: Teilung gemäß Buddy-Strategie, aber Belegung nur der wirklich benötigten Knoten**

Prozessorvergabe - Multiprozessoren: Kern-Fäden

Simulationsergebnisse für die vorangehenden Plazierungsstrategien

- exponentielle Verteilung der Zwischenankunftszeiten und Bedienzeiten
- Höhen und Längen der angeforderten Teilbereiche unabhängig gleichverteilt



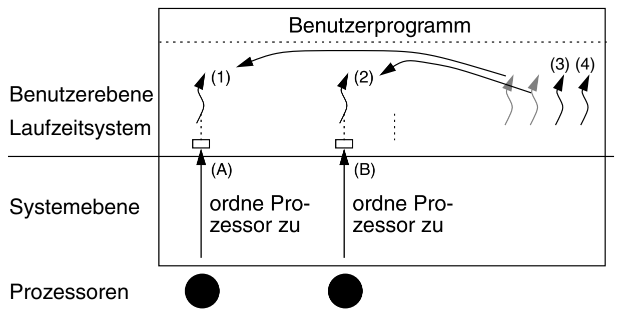
Prozessorvergabe - Multiprozessoren: Kern-Fäden

- **Hand-off Scheduling**
  - **Benutzer gibt Hinweise**
    - Hinweise auf Unzweckmäßigkeit des Zuordnens
    - Hinweise auf zuzuordnende Prozesse

  Letzteres ist insbesondere im Zusammenhang mit Kooperation von Interesse.

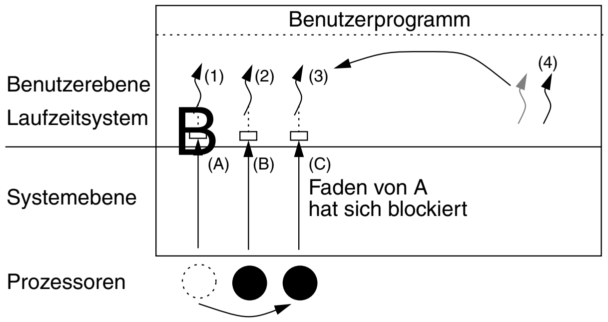
Prozessorvergabe - Multiprozessoren: Kern- und Benutzerfäden

- **Mischung aus Kern- und Benutzerfäden**
  - **Kern ordnet der Anwendung beispielsweise zwei Prozessoren zu**



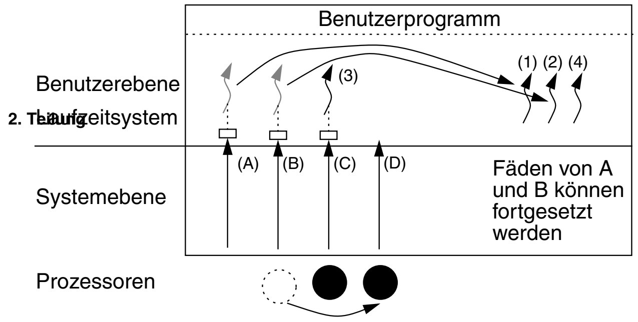
Prozessorvergabe - Multiprozessoren: Kern- und Benutzerfäden

- Ein Benutzerfaden blockiert sich im Kern wegen E/A



Prozessorvergabe - Multiprozessoren: Kern- und Benutzerfäden

- E/A abgeschlossen



Prozessorvergabe - Multiprozessoren: Kern- und Benutzerfäden

- **‘upcall’ wegen freien Prozessors**