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)

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

- 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

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

- 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

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

- 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.

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

- **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

- **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()

- **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);
}
```


- **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.

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.**

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig