Das MEMSY-Programmiermodell

Stefan Turowski

Universität Erlangen-Nürnberg — IMMD IV
SFB 182 TP B2
turo@informatik.uni-erlangen.de

Überblick

- ❑ Grundlegende Überlegungen
- ❑ Architektur
- ❑ Programmiermodell
- ❑ Erweiterungen
- ❑ Zusammenfassung

Anforderungen

Grundlegende Überlegungenl

❏ Abstraktionsgrad

hoch ⟷ niedrig

❏ Anforderungen der Anwendungen

shared memory ⟷ messages

viel Komm. ⟷ wenig Komm.

❏ Implementierungsaufwand

hoch ⟷ gering

❏ Erzielung von Parallelität

explizit ⟷ implizit

Entscheidungen

Schwerpunkte für MEMSY

- Effizienz
- Vielseitigkeit
- Erweiterbarkeit
- vertretbarer Implementierungsaufwand
- Portierbarkeit
- Simulation

Architektur

- ❏ Hardwaresicht
  - Speicher
  - Knoten
- ❏ Applikationen
  - Starten
  - Konfigurieren
- ❏ Programmiersprachen
  - C und C++
    - einfachere Implementierung
    - bessere Compiler
    - effizienterer Code
- ❏ Simulator

Programmiermodell

- explizite Verwaltung des Kommunikationsspeichers durch die Anwendungen
- Adressierung der Knoten relativ über Himmelsrichtungen oder absolut
- Angebot an verschiedenen Kommunikations- und Koordinierungsmechanismen
  - Shared Memory
  - Messages
  - Signale
  - Spinlocks
  - Semaphore
  - Transport

Mechanismen I

- Shared Memory
  - Create + Attach Mechanismus (Segmente)
  - Zugriff über Zeiger (C!)
  - dynamisches Erzeugen und Vernichten von Segmenten
  - Cache Kontrolle
  - (Swapping)
- Messages
  - kurze (2 Worte) Nachrichten
  - blockierend oder nicht blockierend
  - Absender kann festgestellt werden
  - für Koordinierungszwecke
- Signale
  - Unterbrechungen zur Behandlung von Ausnahmesituationen

Mechanismen II

- ❑ Spinlocks
  - Koordinierungsvariablen im Kommunikationsspeicher
  - implementiert durch spezielle Prozessorinstruktion (XMEM)
  - “busy-wait” - für kurze kritische Abschnitte
  - timeout - Mechanismus
- ❑ Semaphore
  - globale Koordinierungsvariablen
  - implementiert durch das Betriebssystem
  - kein “busy-wait”
- ❑ Transport
  - effizienter Transport von größeren Datenmengen
  - implementiert durch das Betriebssystem
  - dynamische Auswahl der besten Alternative

andere Funktionen

- ❑ Informationsmechanismen
  - Prozessoridentifikation und Position im Feld
  - Systemzustand
- ❑ I/O
  - normale UNIX-I/O (fopen, fread, fwrite, ...)
  - knotenlokale Datenbereiche
  - globale Datenbereiche
- ❑ Parallelität
  - mfork (Variante von UNIX fork)

Erweiterungen

- ❏ Transportmechanismus
  - Implementierung
  - Einflußnahme durch Anwendungen
- ❏ Feedback der Anwender
- ❏ Analyse von Anwendungen

laufende Arbeiten

- ❑ Strömungsmechanik
- ❑ Theoretische Chemie
- ❑ Raytracer
- ❑ Testprogramme

Zusammenfassung
q

- ❏ flexibles und architekturnahes Programmiermodell
- ❏ UNIX-Entwicklungsumgebung
- ❏ Standardprogrammiersprachen