Programmiermodelle für verteilten Speicher

Verteilte Softwaresysteme

Prof. Dr. Oliver Braun

Letzte Änderung: 13.11.2018 09:51

Verschiedene Modelle

- nebenläufige Modelle
  - nachrichtenbasierte Modelle
  - auf Datenparallelität basierende Modelle
- kooperative Modelle
  - nachrichtenbasierte Modelle
  - entfernte Aufrufe

Nebenläufige Modelle

Nachrichtenbasierte Modelle

- bei Threads wird über einen gemeinsamen Speicher synchronisiert
- Locks stellen z.B. sicher, dass nur ein Thread schreiben kann
- Locks können aber zu Deadlocks führen
- nachrichtenbasierte Modelle haben *keinen* gemeinsamen Speicher
- es gibt nur Primitive für das Senden und Empfangen von Nachrichten
- daher sehr gut geeignet für verteilten Speicher (und Senden über ein Netzwerk)
- im Folgenden sehen wir uns die wichtigsten Vertreter an

Message Passing Interface (MPI)

- Nachrichtenaustauschbibliothek
- ermöglicht auf homogenen Parallelrechnern effiziente und schnelle Kommunikation
- heute wohl am meisten eingesetztes Modell für parallele und nebenläufige Programmierung
- Broadcast-Anweisung mit der ein Prozess an alle anderen senden kann
- Barrieresynchronisation, bei der die Prozesse warten, bis alle eine Barriere erreicht haben
- ab MPI-2 auch dynamische Prozesse

Occam

- Assembler für *Transputer*
- Transputer: RISC-Prozessor mit vier bidirektionalen Verbindungskanälen
- über diese können Transputer zusammengeschlossen werden, z.B.
  - 2 $\Rightarrow$ Pipe, 3 $\Rightarrow$ Baum … Hypercube
- zu speziell, kein Betriebssystem, wie Unix oder DOS, daher nicht auf Desktop
- für Microcontroller und Embedded Systeme zu mächtig und zu teuer
- Occam baut auf **Communicating Sequential Processes (CSP)** auf
- der Transputer ist Geschichte aber nicht Occam
- Occam gibt es für andere Plattformen und (z.B. Intel) und Java-Implementierungen

PVM (Parallel Virtual Machine)

- PVM ermöglicht mehrere Unix- oder Windows-Rechner zu einer virtuellen Maschine zusammenzufassen
  - parallele Recheneinheit mit verteiltem Speicher
- entwickelt von
  - University of Tennessee, Oak Ridge National Laboratory und Emory University
- Bibliotheken für C, Fortran, Java, Perl
- objektorientierter Aufsatz für C++: CPPVM
- Weiterentwicklung und Verschmelzung mit MPI
  - Heterogenous Adaptable Reconfigurable Networked Systems (HARNESS)
- Hauptziel von PVM ist nicht möglichst hohe Rechenleistung um jeden Preis
  - Möglichkeit verschiedene Hardware, Architekturen und Betriebssysteme zu einem heterogenen Cluster zusammenzuschließen

Funktionale Programmiersprachen

- funktionale Sprachen wie Haskell, Erlang, Scala, ...
- Grundidee der imperativen (von-Neumann-)Sprachen:
  - Zustandsänderung, z.B. Schleifen
- Grundidee der funktionalen Programmierung:
  - unveränderbare Werte, Funktionen berechnen Ergebnisse ohne Zustandsänderung, Rekursion
- *rein funktionale* Funktionen haben keine Seiteneffekte und liefern bei gleichen Parametern immer das selbe Ergebnis
  - **Referentielle Transparenz**
- dadurch leicht(er) parallelisierbar

Datenparallelität ausnutzende Modelle

- **High Performance Fortran (HPF)**
  - der HPF-Compiler bietet einen globalen Adressraum
  - Verteilung der Daten durch Direktiven
  - Kommunikationsroutinen zum Zugriff werden automatisch generiert
- **High Performance Java (HPJava)**
  - paralleles Programmieren auf verteilten und gemeinsamen Speichern
  - ähnlich zu HPF
  - HPJava erweitert Java um multidimensionale Felder (multiarray) ähnlich den Feldern in HPF
  - seit 2007 ruht das Projekt

Kooperative Modelle

Lokalisierung des Kooperationspartners

1. **statisches Binden**
   - durch direkte Angabe der Adresse
   - Aufruf wird zur Übersetzungszeit eingebunden
   - Ändert sich etwas, müssen die Programme neu compiliert werden
   - oder über Config-Datei
2. **dynamisches Binden**
   - Umsetzen eines logischen Namens in physikalische Adresse
   - über Broadcast oder Broker
   - dynamische Systemrekonfigurationen möglich

Broker

- Dienst muss Namen, Versionsnummer, … und Adresse an Broker senden
- Server **registriert** den Dienst beim Broker
- Eintrag muss vom Dienst wieder aktiv **deregistriert** werden
- Anfrage eines Clients
    1. Client fragt Broker nach Adresse eines Dienstes.
    2. Broker gibt dem Client die Adresse.
    3. Client fragt Dienst direkt an.
    4. Server führt Dienst aus und antwortet dem Client.

Nachrichtenbasierte Modelle — TCP/IP-Sockets

- *Netzwerkprogrammierung* — bekannt aus Netzwerke-Vorlesung
- für **synchrone** Nachrichtenübertragung das am meisten verbreitete System
- de-facto Standard, da von jedem Betriebssystem angeboten
- ursprüngliche Implementierung in 4.3BSD UNIX (DARPA-Auftrag)

Nachrichtenbasierte Modelle — Java Message Service

- Sender und Empfänger über Message-Service nur lose gekoppelt
- d.h. alle Kommunikation über Message-Service
- asynchrone Kommunikation
- JMS:
  - zentraler Message-Server (*JMS Provider*)
  - viele JMS-Clients
- zwei Nachrichtenmodelle (*Message Domains*)
  1. **Point-to-Point**-Modell (1:1-Kommunikation)
  2. **Publish/Subscribe**-Modell (1:m-Kommunikation)

Entfernte Aufruf-Modelle

- nicht verteilte, monolithische Anwendung kann als Ansammlung von Prozeduren betrachtet werden
- Idee: Prozeduren auf verschiedene Rechner verteilen und entfernte Prozeduren aufrufen
- entfernte Prozeduraufrufe
  - **Remote Procedure Calls** (RPCs)
- entfernte Methodenaufrufe (OOP)
  - **Common Object Request Broker Architecture** (CORBA)
  - **Remote Method Invocation** (RMI)
- entfernte Komponentenaufrufe
  - **Distributed Component Object Model** (DCOM)
  - **.NET Remoting**
- entfernte Serviceaufrufe
  - **Web Services**
  - **XML-RPC**