Verteilte und Parallele Datenbanksysteme

Vorlesung

Mo. 11:15-13:00 Uhr (s.t.)
Do. 11:oo-12:00 Uhr (s.t.)

Donald Kossmann
kossmann@in.tum.de

Was ist ein verteiltes Datenbanksystem?

Sammlung von logisch verwandten Daten, die auf mehreren Knoten eines Computernetzwerkes gehalten werden.

1. **mehrere Knoten:** Abgrenzung zu zentralen Systemen
2. **logisch verwandt:** es gibt eine globale Anwendung (d.h. irgendjemand ist daran interessiert, auf alle Daten zuzugreifen)

Beispiele

1. Bank mit Filialen in verschiedenen Städten

   1. mehrere Knoten: Kontoführung in jeder Filiale

   2. globale Anwendungen: Überweisungen, Geldautomat, Management der Bank, ...

2. World Wide Web

   1. mehrere Knoten: viele WWW Server auf der ganzen Welt

   2. globale Anwendungen: Surfen, Chat, ...

3. Mein Büro

   1. mehrere Knoten: 3 Schreibtische mit jeweils 4 Schubladen
   (Vorsicht das ist kein Computernetzwerk!)

   2. globale Anwendungen: Arbeiten

Schlüsselbegriff: Transparenz

Szenario I: Hölle (keine Transparenz)

Probleme:

1. Anwendungsprogramme zu häcken ist ‘ne Qual
2. Performance ist furchtbar

Szenario II: Himmel (Transparenz)

Anwendungsprogrammierer braucht nur mit dem VDBMS zurecht zu kommen.

Beispiele *Bank, WWW* und *Mein Büro*: Die Wahrheit liegt irgendwo zwischen Himmel und Hölle.

Sieben Arten von Transparenz

1. Location Transparency
2. Performance Transparency
3. Copy Transparency
4. Transaction Transparency
5. Fragment Transparency
6. Schema Change Transparency
7. Local DBMS Transparency

Wieso baut man überhaupt verteilte Systeme

I. Man hat gar keine andere Wahl

A-posteriori Integration von einzelnen Datenbanken ⇒ **heterogene** verteilte Datenbank.

II. Weil es besser ist

Durch mehrere Knoten Hoffnung auf:

1. bessere Performance
2. höhere Zuverlässigkeit
3. einfachere Erweiterbarkeit des Systems
   (einfach Knoten für neue Business-Unit hinzunehmen)

Es entsteht eine **homogene** verteilte Datenbank.

Wieso kann man in einem verteilten System eine bessere Performance erzielen?

Chancen:

1. Man spart Netzwerkkosten
   Halte Daten dort, wo sie gebraucht werden
   (Allokation)
2. Parallelität
   Eine oder mehrere Operationen können auf mehreren Rechnern gleichzeitig ausgeführt werden

Risiken:

1. Netzwerkkosten steigen wegen Synchronisationsaufwandes und schlechter Allokation
2. Keine Parallelität wird erzielt wegen schlechter Lastbalancierung

Wieso ist ein verteiltes System zuverlässiger?

Lege Kopien von wichtigen Daten auf mehreren Knoten an.

Chancen:

1. Ausfall eines Knotens

2. Netzwerkpartitionierung

Kosten und Risiken:

1. Kopien müssen konsistent gehalten werden
2. Manchmal darf niemand Update machen, obwohl Kopie vorhanden

Was ist ein Paralleles DBMS?

- es ist eine besondere Art von verteiltem DBMS
- enge Kopplung der Knoten
- im Vordergrund steht *Ausführung* von Operationen auf mehreren Knoten
  nicht so sehr *Speicherung* von Daten auf mehreren Knoten
- Ziel meist: Performancegewinn durch Parallelität
  oft um den Preis höherer Kommunikation
- implementiert auf sehr schnellen Netzwerken (häufig Spezialnetzwerke)
  Schlagworte: Mehrprozessorrechner, Cluster von Workstations

Wieso sind verteilte und parallele DBMSs gerade heute so interessant?

1. Trend: Netzwerktechnologie ausgereift, Kommunikationskosten sinken

(Dadurch überhaupt erst möglich.)

2. Trend: Commodity Hardware

100 Pentium Chips sind viel billiger als eine Cray und zigmal leistungsfähiger

Kleiner historischer Exkurs

60/70 Jahre: Hardware teuer, Kommunikation teuer

- Aufbau von Rechenzentren
- notgedrungene Zentralisierung der betrieblichen Abläufe
- Batch Processing

80/90 Jahre: Hardware billig, Kommunikation billig

- Dezentralisierung möglich
- on-line processing
- Jeder muß mitmachen, denn die Konkurrenz macht es auch

(Technischer Fortschritt macht die Menschheit nicht unbedingt glücklicher, aber er hilft, damit sie sich nicht so langweilt.)

Aufbau eines verteilten Datenbanksystems

Ziel dieser Vorlesung:

Die einzelnen Mosaiksteinchen beherrschen.

Eure Aufgabe später:

Bestehende Mosaiksteinchen verwenden und anpassen, ggf. eigene bauen und das Ganze mit Uhu zusammenzukleben. (In der Praxis gibt es da nicht viel Spielraum; es sieht aber jedes Mal anders aus.)

Übersicht

1. Einführung
   Motivation, Trends, Begriffe, ...
2. Netzwerke
   Netztopologien, Ethernet, Tokenring, ISO-OSI Modell, TCP/IP, ...
3. Partitionierung und Allokation
   Wie teilt man (globale) Relationen auf?
   Wo speichert man die einzelnen Teile?
4. Anfragebearbeitung
   Reduktion von Anfragen, Anfrageoptimierung (Join Ordering, Site Selection), Kostenmodelle, Semi-Join Verfahren
5. Ökonomische Modelle
   Verteiltes System = Marktplatz, Knoten = Marktstand
6. Verteilte Transaktionsverwaltung
   kurze Übersicht, 2-Phase Commit, verteilte Deadlockerkennung, geschachtelte Transaktionen
7. Replikation
   Was repliziert man?
   Wie hält man die Kopien konsistent?
8. Parallele Datenbankmaschinen
   Architekturen, Formen von Parallelität, Parallelisierung von relationalen Operatoren

9. Client-Server Architekturen
   SAP R/3, Marktplätze, publish & subscribe, push vs. pull, Namenserver, Data Warehouses
10. Heterogene verteilte Systeme/Interoperabilität
    Datenmodell, Wrapper Architektur, Einschränkungen bei der Anfragebearbeitung
11. XML und Datenbanken
12. WWW und Suchmaschinen
13. Zusammenfassung

Literatur

1. P. Dadam: Verteilte Datenbanken und Client/Server-Systeme (Kapitel 1-10, 12)
   stark angelehnt an Ceri-Pelagatti; sehr gut
   Hierzu gibt es auch Folien im Web.

2. S. Ceri, G. Pelagatti: Distributed Databases – Principles & Systems (Kapitel 1-10)
   (der Klassiker)

3. T. Özsu, P. Valduriez: Principles of Distributed Database Systems
   wie 1 und 2, aber ich mag es nicht so sehr

4. A. Kemper, A. Eickler: Datenbanksysteme – eine Einführung (Kapitel 15)
   sehr schöne Übersicht zum Warmwerden

5. D. DeWitt, J. Gray: Parallel Database Systems: The Future of High Performance Database Systems
   Artikel in CACM 1992
   Klassiker zum Thema "parallele Datenbanksysteme"

6. Weitere Artikel aus der Forschungsliteratur
   (läßt sich leider nicht vermeiden)