AK-Treffen „Mobile Datenbanken und Informationssysteme“

Eine Architektur für dynamische Replikation in mobilen Datenbanksystemen

Christoph Gollmick
gollmick@informatik.uni-jena.de

Friedrich-Schiller-Universität Jena
Institut für Informatik
Lehrstuhl für Datenbanken und Informationssysteme

Jena, Thüringen *23. November 2001*

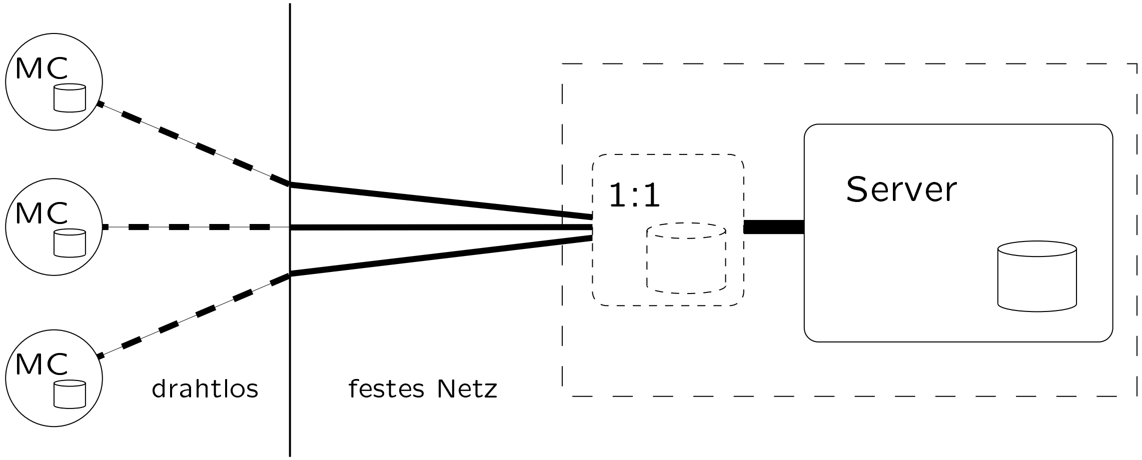
Gliederung

1. Einführung
2. 3-Tier Architektur
3. Problemkreise
   Datenhaltung, Reintegration, Mobilität, ...
4. Zusammenfassung und Ausblick

Anforderungen an mobilen Datenbankzugriff

- client-orientiert
  - Berücksichtigung der Beschränkungen mobiler Clients
  - Unterstützung lokaler Änderungsoperationen
- Infrastruktur
  - hohe Skalierbarkeit (Tausende mobile Clients)
  - Anpassung an Mobilitätsprofil der Clients
  - Anpassung an sich ändernde Zugriffsprofile
  - Minimierung von Verbindungszeiten und -kosten
  - Ausfalltoleranz

Klassische Architektur - erweitertes C/S-Modell

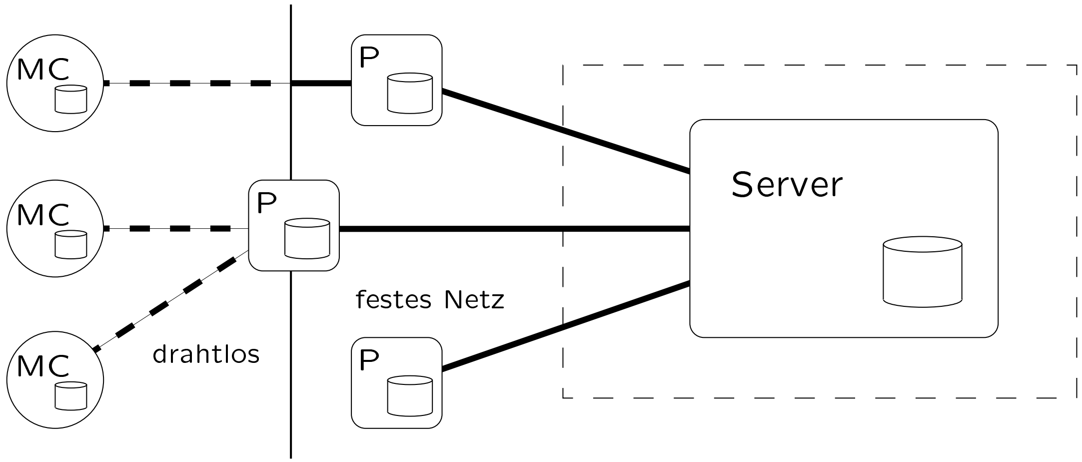


- zentral definiertes Replikationsschema und -verfahren
- client- und anwendungsorientiert
- Skalierbarkeit nur durch Einsatz besserer/zusätzlicher Hardware
- lange Kommunikationswege

→ Replikate von Daten dort verwalten, wo sie gebraucht werden

→ Kombination von Replikation mit Proxy-Caching (dynamische Replikation)

3-Tier-Architektur für mobilen Datenbankzugriff



- Proxy-Server im festen Netzwerk
- Proxies sind geographisch verteilt (Zuständigkeitsbereiche)
- Clients kommunizieren i.d.R. über nächstgelegenen Proxy-Server
- Ausnutzung der Lokalität im Client-Zugriffsprofil
- Proxies replizieren entsprechende Teilmenge der Server-Daten

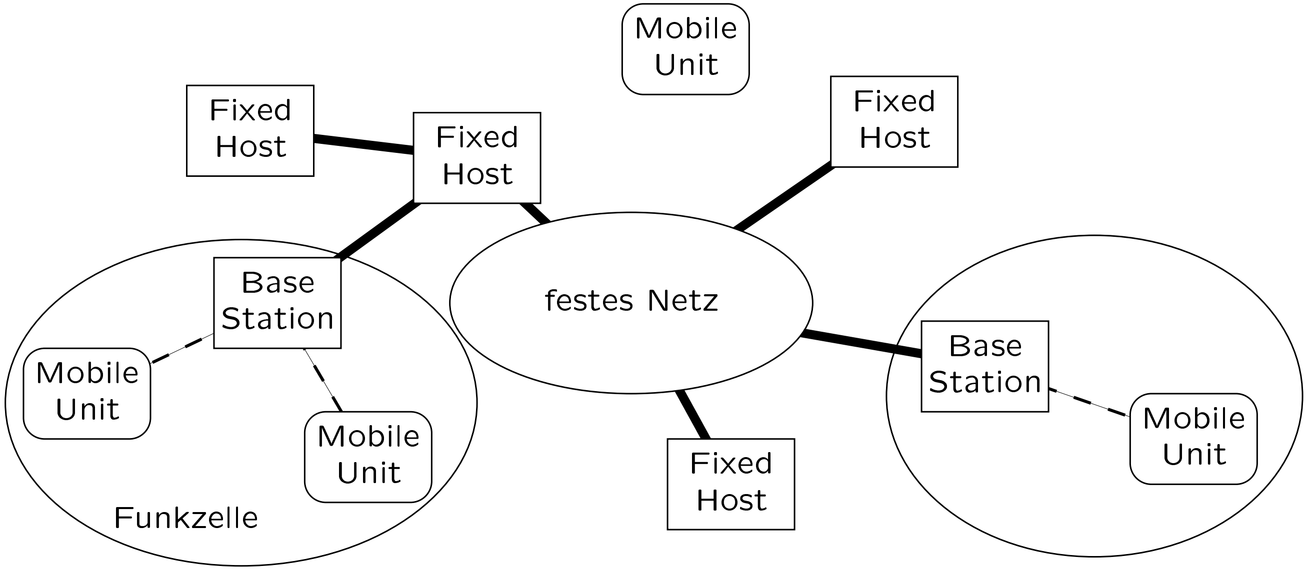
Problemkreise

- Datenhaltung im Proxy
- Abbildung auf physische Strukturen
- Unterstützung der Reintegration
- Unterstützung der Mobilität
- Weitere hier nicht betrachtete Problemkreise:
  aktive Elemente, Heterogenität, Zugriffsschutz, ...

Datenhaltung im Proxy

- Replikatverteilung ergibt sich durch Anfragen der Clients
- Möglichkeiten der Speicherung
  - Cache für Anfrageergebnisse (materialisierte Sichten)
  - Cache der Rohdaten
- verschiedene Granulate
  - physische (z.B. Seiten)
  - logische (z.B. einzelne Tupel, Tabellen, Fragmente)
- zunächst Unterscheidung in
  - client-spezifische Daten $\rightarrow$ getrennte Caches
  - gem. genutzte Daten $\rightarrow$ *ein* Cache plus Indexunterstützung
- Optimierungen (z.B.)
  - dynamische Partitionierung
  - Hauptspeicherdatenbanken

Abbildung auf physische Strukturen – Beispiel



- Abbildungen

Server	$\rightarrow$	*Fixed Host*
Proxy	$\rightarrow$	*Fixed Host* oder *Base Station*
mobiler Client	$\rightarrow$	*Mobile Unit*

- Einsatz vorhandener Hardware und Strukturen möglich
- Ausnutzung einer Proxy-Hierachie

Unterstützung der Reintegration

- wesentliches Anliegen, am Bsp. *Two-Tier Replication* (PCs in Proxies)
- Ziel: lokales Einbringen und lokale Konfliktauflösung
- Grundlagen
  - Statistiken über Zugriffsverhalten
  - Bildung von Replikatmengen, deren Primärkopien
    - auf genau einem Proxy und
    - so nah wie möglich an unterstützten Clients liegen
- bei Optimierung Unterscheidung in
  - ortsabhängige Daten → Proxy im Zuständigkeitsbereich
  - ortsunabhängige Daten
    - client-spezifische Daten → nächstgelegener Proxy
    - gem. genutzte Daten → Kostenmodell für optimalen Proxy
- Optimierung muß weitere Kriterien berücksichtigen
- Proxy-Hierarchien begrenzen an Reintegration beteiligte Proxies

Unterstützung der Mobilität

- mobile Clients wechseln Zuständigkeitsbereiche (Proxies)
- replizierte Daten eines Client werden in neuen Proxy übernommen
- Entscheidungmöglichkeiten für Primärkopie
  - Änderungen weiterleiten
  - Anpassung von Replikatmengen und Transfer von Primärkopie
- Konsequenzen nach Datenarten:
  - ortsabhängige Daten $\rightarrow$ immer weiterleiten
  - ortsunabhängige Daten
    - client-spezifische Daten $\rightarrow$ einfaches Kostenmodell
    - gem. genutzte Daten $\rightarrow$ Kostenmodell für mehrere Clients
- nur lokale Statistiken und Optimierung sinnvoll
- Antizipieren der Bewegung, vorausschauendes Handeln

Zusammenfassung und Ausblick

Vorteile 3-Tier-Architektur

- Verkürzung von Synchronisationszeiten
- lokale Begrenzung von Änderungskonflikten
- Verringerung der Netzbelastung
- Skalierbarkeit
- Ausfalltoleranz und Zugriffsschutz

Ausblick

- Algorithmen für dynamische Replikation (Statistik, Optimierung)
- Datenhaltung im Proxy (Fragmentierung, Indizierung)
- verteilte Ausführung aktiver Elemente (z.B. zur Integritätssicherung)
- prototypische Implementierung und Simulation