Serverseitige Auswertung von Indexen semantischer, clientseitiger Caches in mobilen Informationssystemen

Hagen Höpfner*
hoepfner@iti.cs.uni-magdeburg.de
Institut für Technischen und Betriebliche Informationssysteme
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
Postfach 4120, 39016 Magdeburg, Deutschland

Abstract: Die Probleme drahtloser Datenübertragung motivieren das Cachen von Daten auf Mobilgeräten. Diese unterliegen aber starken Hardwarerestriktionen. Die Komplexität der Algorithmen zum Verwalten und Ausnutzen von Caches übersteigt daher häufig die Fähigkeiten der Mobilgeräte oder führt zu schlechten Antwortzeiten. Im Rahmen dieses Beitrags werden daher erste Ideen für die Migration von Berechnungen im Zusammenhang mit clientseitigen, semantischen Caches vom mobilen, leichtgewichtigen Client zum stationären, leistungsfähigen Server vorgestellt.

1 Einleitung und Motivation

Ein zentrales Problemfeld mobiler Informationssysteme (IS) bildet die Anbindung mobiler Endgeräte über Funknetzwerke. Datenübertragung über GSM/GPRS ist verhältnismäßig teuer und langsam, wohingegen schnellere, preiswerte Netze nicht flächendeckend verfügbar sind. Ein Ansatz, der dieses Problem adressiert ist das clientseitige Cachen von Anfrageergebnissen. Hierzu kommen oftmals semantische Caches zum Einsatz, welche die Anfragen nutzen, um die entsprechenden Ergebnisse lokal zu indexieren. Bei neuen Anfragen kann somit die vom Server anzufordernde Datenmenge reduziert werden. Leider ist das Testen von semantischen Überlappungen rechenintensiv. Insbesondere durch die Trends im Handysektor werden aber zunehmend leichtgewichtige Endgeräte als Clients von IS eingesetzt. Da sie aber neben der Cacheverwaltung auch Applikationen bereitstellen müssen, ist es sinnvoll, den Aufwand, der für die Berechnung von wiederverwendbaren Daten notwendig ist, zu minimieren. In der vorliegenden Arbeit wird untersucht, ob eine Berechnungsmigration auf den Server machbar und wann sie sinnvoll ist.

2 Grundlagen

Die in dieser Arbeit vorgestellten Untersuchungen basieren auf eigenen Vorarbeiten aus dem Bereich des semantischen Cachens [HS03b] sowie der serverseitigen Unterstützung

*Gefördert durch die DFG unter der Fördernummer SA 782/3-2.

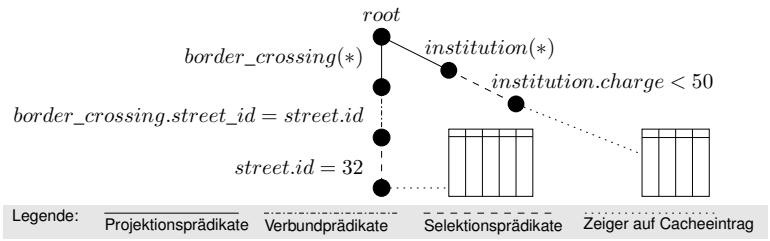

Abbildung 1: Clientseitiger Cacheindex

von einer großen Anzahl von Mobilgeräten [HS03a, HSS04]. Anfragen werden in Form von konjunktiv verknüpften Prädikaten gestellt, wobei zwischen drei[1] Arten von Prädikaten unterschieden wird. *Projektionsprädikate*[2] sind vergleichbar mit dem Projektionsoperator der Relationenalgebra. *Verbundprädikate* dienen dem Verbund von zwei Relationen und entsprechen dem θ-Verbund der Relationenalgebra. *Selektionsprädikate* repräsentieren Tupelmengenbeschränkungen analog zum Selektionsoperation der Relationenalgebra. Mithilfe einer lexikographischen Ordnung innerhalb dieser Prädikatklassen werden Anfragen nun als Prädikatsequenzen gestellt, wobei zuerst die Projektions-, dann Verbund- und abschließend Selektionsprädikate aufgeführt werden. Sowohl client- als auch serverseitig werden Anfragen in Form von Pfaden jeweils eines Tries [Fre59] gespeichert, wobei auch alternative Repräsentationen denkbar wären. Auf dem Mobilgerät dient dieser Trie als Index des semantischen Caches. Abbildung 1 illustriert den Trie-basierten semantischen Cache. In diesem Beispiel hat der Client zwei Anfragen mit unterschiedlichen Projektionsprädikaten gestellt, wobei die erste (links abgebildet) sowohl ein Verbund- als auch ein Selektionsprädikat benutzt. Die zweiten Anfrage enthält kein Verbundprädikat.

Serverseitig wird der Trie benutzt, um effizient festzustellen, welche Clients welchen Datenbereich im Cache halten. Dies ist für die Benachrichtigung der Clients bei Aktualisierungen des Serverdatenbestandes aus Gründen der Cachekohärenz notwendig. Abbildung 2 illustriert beispielhaft den Aufbau dieses Anfrageindexes, wobei auffällt, dass hierbei keine Daten, sondern vielmehr die IDs derjenigen Clients indexiert werden, die die mit den Anfragen korrelierten Daten im lokalen Cache halten. Die Anfragen des oben genannten Clients spiegelt sich somit auch serverseitig wieder. In diesem Beispiel kann festgestellt werden, dass der entsprechende Client die ID 23 besitzt.

3 Berechnung von Kompensations- und Filteranfragen

Die Idee des semantischen Caching ist, dass eine neue Anfrage Q mit der Menge der bereits gestellten Anfragen $\mathcal{P} = \{P_1, \ldots, P_n\}$ verglichen wird und gegebenenfalls deren Ergebnisse wiederverwendet werden. Dabei können die folgenden Fälle auftreten:

[1] Die in [HS03a] angesprochenen Kontextprädikate werden hier vorerst nicht betrachtet.
[2] in Vorarbeiten als Relationsprädikate bezeichnet

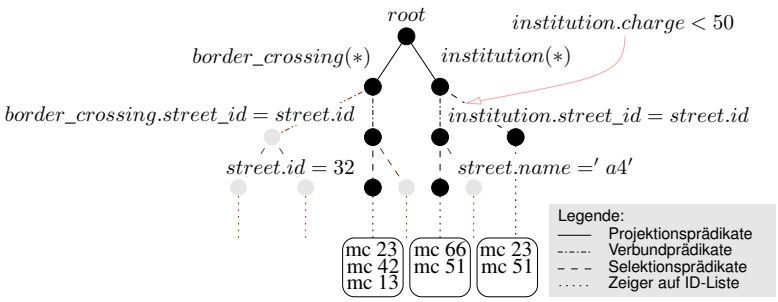

Abbildung 2: Serverseitiger Anfragebaum

Identität: Es existiert eine identische Anfrage P_i mit $1 \leq i \leq n$ im Cache. Damit kann Q komplett lokal beantwortet werden.
Inklusion: Der von Q abgedeckte Wertebereich ist vollständig in dem von $\mathcal{P}$ abgedeckten enthalten. Damit kann Q vollständig lokal beantwortet werden.
Überlappung: Die von $\mathcal{P}$ und Q abgedeckten Wertebereiche überlappen, sodass Q partiell lokal beantwortet werden kann. Für die fehlenden Daten wird jedoch eine Kompensationsanfrage benötigt (vgl. [HS03b]).
Disjunktheit: Q kann nicht einmal partiell lokal beantwortet werden, da weder Identität noch Inklusion oder Überlappung vorliegen.

Festzuhalten bleibt, dass Disjunktheit und Überlappung garantiert einer Kommunikation mit dem Server bedürfen. Im Gegensatz dazu könnten Identität und Inklusion komplett lokal behandelt werden. Aufgrund der Anfragestruktur und der entsprechenden Ordnung der Prädikate innerhalb von Anfragen kann die Identität von Q und P_i recht einfach mittels paarweisen Vergleichens der Prädikate in P_i und Q nachgewiesen (oder widerlegt) werden, sodass dieser Fall problemlos lokal geprüft werden kann. Das zentrale Problem, das diesem Papier zu Grunde liegt, ist aber der Aufwand (z.T. und unter entsprechenden Beschränkungen, kubischer Zeitaufwand bezogen auf die Anzahl verwendeter Variablen), um bei konjunktiven Anfragen festzustellen, welcher der anderen drei Fälle vorliegt. Es gibt zahlreiche Arbeiten, die die Berechnungskomplexität von entsprechenden Prädikatvergleichen untersucht haben. Zusammenfassungen hierzu sind u.A. in [RD98] und [Tür99] zu finden.

Wie bereits erwähnt ist es die Idee, diese Berechnung an den Server zu delegieren. Daraus resultiert auf der Clientseite folgendes Vorgehen: Zuerst wird geprüft, ob für Q im Cache bereits eine identische Anfrage enthalten ist. Ist dies der Fall, wird Q aus dem Cache beantwortet. Anderenfalls wird Q zusammen mit der entsprechenden Client-ID an den Server geschickt.

Auf dem Server muss nun zuerst geprüft werden, welche Anfragen der Client gestellt hat und somit, welche Daten dessen Cache enthält. Um dies effizient zu realisieren ist eine Adaption des serverseitigen Anfrageindexes notwendig. Hierfür wird eine sortierte Liste eingesetzt, deren Elemente auf die Blattknoten des Tries verweisen (vgl. Abbildung 3). Somit können durch bottom-up Traversierung des Tries effizient die Anfragen ermittelt

werden, deren Ergebnisse im entsprechenden Clientcache vorliegen.

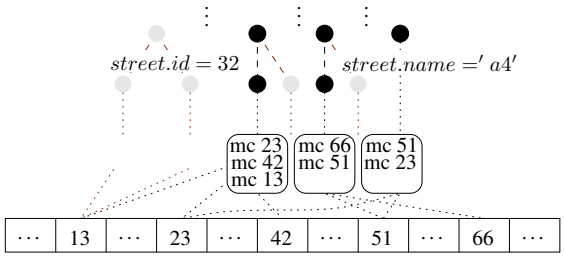

Abbildung 3: Erweiterter serverseitiger Anfragebaum

Anschließend ermittelt der Server die Art der Korrelation zwischen der neuen Anfrage und den gecachten Anfragen. Das weitere Vorgehen ist wie folgt:

Disjunktheit: Wurde festgestellt, dass der Client keine wiederverwendbaren Daten enthält, d.h., der Server konnte keine Anfrage im Trie finden, die mit Q überlappt, wird Q im Server-Trie eingetragen und das Anfrageergebnis an den Client übertragen. Dieser speichert das Ergebnis und passt seinen Cacheindex entsprechend an.
Überlappung: Wurde festgestellt, dass der Client eine Teilmenge des Ergebnisses bereits enthält, so wird die entsprechende Kompensationsanfrage K erzeugt, ausgeführt und zusammen mit dem entsprechenden Anfrageergebnis an den Client übertragen. Auf diesem kann nun Q lokal ausgeführt und mit dem Ergebnis vom K vereinigt werden. Zusätzlich wird das Ergebnis von K auf dem Client gecacht.
Inklusion: Wurde festgestellt, dass der Client eine Obermenge des Ergebnisses enthält, so wird dieser darüber informiert, dass er Q lokal beantworten kann. Ein Eintrag der Filteranfrage ist hierbei weder auf der Client- noch auf der Serverseite notwendig.

4 Szenarienspezifische Nutzbarkeit

Eine genaue Diskussion eines Kostenmodells würde den Rahmen dieses Papiers sprengen, weshalb wir im Folgenden nur anhand der übertragenen Daten zeigen, in welchen Szenarien die Migration sinnvoll ist.

Im Fall von einer clientseitigen Berechnung ist bei Identität und Inklusion keine Kommunikation notwendig. Bei Überlappung müssen die Kompensationsanfrage (s_K Bytes) und das Ergebnis dieser (s_{EK}) übertragen werden. Im Fall der Disjunktheit werden die s_Q Bytes der Anfrage und s_{EQ} Bytes für das entsprechende Ergebnis übertragen. Die Gesamtübertragung[3] ergibt sich also mit $S^c = \alpha_U * (s_K + s_{EK}) + \alpha_D * (s_Q + s_{EQ})$. Im Fall der Berechnungsmigration kommen zu diesem Wert noch s_Q Bytes für die Anfrage bei einer Inklusion und eine Byte für die entsprechende Rücknachricht. Damit ergibt sich die

[3] $\alpha_U + \alpha_D + \alpha_I + \alpha_A = 1$ sind die Wahrscheinlichkeiten, dass eine Anfrage mit den (bzw. einer der) gecachten Anfragen überlappt (U), disjunkt (D) oder identisch (A) ist, oder dass eine Inklusion (I) auftritt.

Gesamtzahl der bei der Migration zu übertragenden Bytes als $S^m = S^c + \alpha_I * (s_Q + 1)$.

Anhand dieser Formel wird offensichtlich, dass Szenarien, in denen häufig Inklusionen auftreten (großes α_I) nicht geeignet sind, um die Korrelationsberechnung vom Client auf den Server zu verlagern. Eine typische Anwendung ist das folgende Hoarding-Szenario: Abhängig von der Lokation werden *alle* relevanten Daten auf das Mobilgerät übertragen wo sie angefragt werden können. Das heißt, dass ein großer Datenbereich zur Verfügung steht und Anfragen sehr wahrscheinlich aus diesem beantwortet werden können.

Ist hingegen die Wahrscheinlichkeit der Inklusion gering (kleines α_I), macht eine Migration Sinn, wenn die Berechnung auf dem Mobilgeräte zu lange dauern würde (z.B. auf Handys). Ein derartiges Szenario ist z.B. ein Kinoplaner. Hier kommt es häufig zu Überlappungen, da z.B. Filmbeschreibungen aus den Daten der Vorwoche wiederverwendet werden können. Inklusionen sind hingegen unwahrscheinlich. Schließlich interessiert in der Regel nicht, wann in der letzten Woche ein Film gelaufen ist.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurden erste Ideen zur Migration von Berechnungen, die im Zusammenhang mit semantischen Caches auftreten, vom mobilen Client hin zum stationären Server vorgestellt. Es kann hervorgehoben werden, dass es im Rahmen der zugrundeliegenden Infrastruktur möglich ist, den Client zu entlasten, indem davon ausgegangen wird, dass Anfragen, die nicht identisch zu bereits zwischengespeicherten Anfragen sind, an den Server delegiert werden. Der Mehraufwand an Kommunikation beschränkt sich hierbei auf den Fall, dass Filteranfragen zur lokalen Aufbereitung notwendig sind.

In Folgearbeiten muss nun untersucht werden, wann eine Inklusion effizient auf dem Client erkannt werden kann. Des Weiteren muss evaluiert werden, wie groß der zusätzliche Aufwand für den Server ist. Ebenfalls wurde vernachlässigt, dass der Cache einer Ersetzungsstrategie unterliegt und diese sich sowohl clientseitig als auch serverseitig auswirkt.

Literatur

[Fre59] E. Fredkin. *Trie memory*. Information Memorandum, Bolt Beranek and NewMan Inc., Cambridge, MA, 1959.

[HS03a] H. Höpfner and K.-U. Sattler. SMoS: A Scalable Mobility Server. In *Poster Proc. of BNCOD20, Coventry, UK*, pages 49–52. Coventry University, July 2003.

[HS03b] H. Höpfner and K.-U. Sattler. Towards Trie-Based Query Caching in Mobile DBS. In *Post-Proc. of the Workshop Scalability, Persistence, Transactions - Database Mechanisms for Mobile Applications*, LNI, pages 106–121, 2003.

[HSS04] H. Höpfner, S. Schosser, and K.-U. Sattler. An Indexing Scheme for Update Propagation in Large Mobile Information Systems. In *Proceedings of the EDBT-Workshop on Pervasive Information Management, 18. March 2004, Heraklion - Crete, Greece*, pages 31–42, 2004.

[RD98] Q. Ren and M. H. Dunham. Semantic Caching and Query Processing. Technical Report 98-CSE-04, Department of Computer Science and Engineering, Southern Methodist University, Dallas, May 1998.

[Tür99] C. Türker. *Semantic Integrity Constraints in Federated Database Schemata*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 1999.