Kontextsensitive Informationsfilter

Peter Fischer Donald Kossmann

Institut für Informatik
Universität Heidelberg
Im Neuenheimer Feld 348
69120 Heidelberg
{peter.fischer, kossmann}@informatik.uni-heidelberg.de

Abstract: Die Kombination von kontextsensitiven Informationssystemen mit leistungsfähigen Informationsfiltern (Publish/Subscribe) verspricht qualitativ hochwertige individuelle Informationsversorgung bei gleichzeitig guter Skalierbarkeit. Die besondere Herausforderung dabei ist das Zusammentreffen von zahlreichen Kontextänderungen und hohen Nachrichtenraten. Das von uns entwickelte Verfahren verbessert bestehende Indexmethoden dahingehend, sich auf hohe und schwankende Updateraten automatisch anzupassen.

1 Einleitung

Aus der Kombination von kontextbasierten Systemen[De01] und skalierbaren Informationsfiltern ergeben sich bessere Profile für Informationsfilter und bessere Skalierbarkeit für kontextbasierte Systeme, da erstere die automatische Anpassung der Information an die Situation des Benutzers ermöglichen, während letztere in der Lage sind, mit vielen Profilen und hohen Nachrichtenraten effizient umzugehen.

Allerdings verwenden diese beide Konzepte gegensätzliche Ziele und Methoden: bei kontextbasierten Systemen ist mit hohen oder schwankenden Updateraten zu rechnen, während Informationsfilter spezielle Indizes verwenden, die für eine geringe Raten an Updates optimiert sind.

Um diesen Ziel- und Methodenkonflikt zu lösen, schlagen wir ein Verfahren auf der Basis von automatischer und schrittweiser Anpassung vor. Dies ermöglicht es, die bestmögliche Balance zwischen Updatekosten und Kosten für Nachrichtenverarbeitung zu erreichen, ohne dass manuell eingegriffen werden muss; ebenso werden Änderungen im Updateverhalten berücksichtigt. Ein solches Verfahren bedingt einen gewissen Overhead, allerdings zeigen unsere Ergebnisse, dass dieser Overhead relativ klein ausfällt, in einem weiten Bereich unser Verfahren sogar die beste Leistung bietet.

2 Problemstellung

Ein kontextsensitiver Filter muss in der Lage sein, bei einer großen Anzahl von kontextbasierten Profilen eine hohe Rate an Nachrichten zu verarbeiten, wobei sich die Kontextdaten (zumindest zum Teil) schnell ändern. Dabei muss das Ergebnis korrekt sein, sodass keine Nachrichten aufgrund von verzögert berücksichtigten Kontextänderungen inkorrekt versandt oder zurückgehalten werden.

2.1 Definitionen:

Während im Rahmen der Forschung zu kontextsensitiven Systemen zahlreiche verschiedene Definitionen für „**Kontext“** entwickelt wurden, verwenden wir eine allgemein gehaltene Variante, in der Kontexte Attribut-Wert-Paare sind, die einer Person oder einem System zugeordnet sind. Interpretation und Methoden zur Generierung dieser Kontextdaten wurden in anderen Arbeiten hinreichend behandelt; sie sind orthogonal zu den hier vorgeschlagenen Konzepten.

Ebenso bestehen **Nachrichten** aus einer Menge Attribut-Wert-Paare. **Profile** wiederum sind relativ einfache Anfragen, die die Interessen eines Benutzers spezifizieren. Diese Anfragen bestehen aus der disjunktiven Normalform von Vergleichsoperationen. Verglichen werden können dabei jeweils Konstanten, Werte von Nachrichtenattributen und Werte von Kontextattributen, als Vergleichsoperatoren betrachten wir =, >,<; der Ansatz ist jedoch auch auf Mengeninklusion oder räumliche Strukturen anwendbar.

2.2 Anwendungen

Zwei Beispiele von Nutzungsmöglichkeiten sind a) Messagebroker mit Feedback, bei den der Zustand der Zielsysteme (Load, Verfügbarkeit von Ressourcen) die Routingentscheidung beeinflusst und b) verallgemeinerte ortsbezogene Dienste, bei denen nicht nur die direkte Ortsinformation, sondern auch Eigenschaften des Benutzers oder des Ortes die Informationsauswahl beeinflussen. In beiden Fällen ist mit vielen Kontextänderungen zu rechnen, bei a) auch mit einer sehr hohen Nachrichtenrate, bei b) dagegen mit sehr vielen Profilen.

3 Architektur eines kontextsensitiven Filters

In Abbildung 1 geben wir einen Überblick über die Architektur und die Komponenten eines kontextsensitiven Informationsfilters, wobei nicht jede Variante tatsächlich alle Komponenten verwendet:

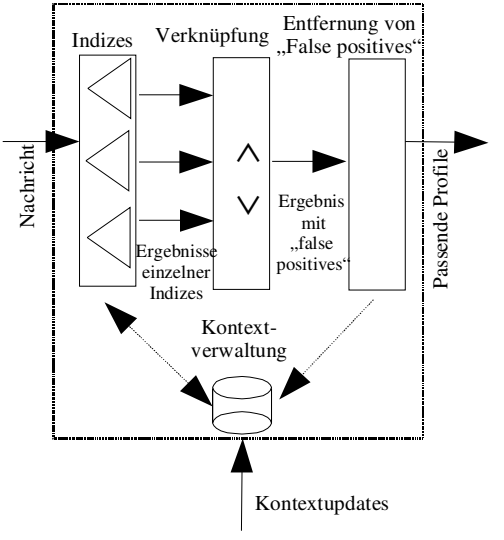

Abbildung 1: Architektur eines kontextsensitiven Informationsfilters

3.1 Kontextverwaltung

Die Kontextverwaltung speichert Daten für Kontextwerte und Konstanten, alle Änderungen von Kontextdaten und Profile werden mindestens hier durchgeführt.

Diese Daten werden von den Indizes und der Entfernung der „False Positives“ benötigt.

3.2 Indizes

Da es die Aufgabe eines Informationsfilters ist, Profile zu finden, für die eine bestimmte Nachricht relevant ist, wurden in Forschung dafür Indizes entwickelt, die diesen Vorgang beschleunigen. Hierbei

ergeben sich eine Reihe von Freiheitsgraden: die Indexstruktur (B-Bäume, R-Bäume, Intervalindizes, Spezialindizes usw.), das Indexziel (Daten, Struktur), die Genauigkeit (exakte Ergebnisse oder false positives) sowie der Bereich, der auf dem indiziert wird (alle Attribute oder ausgewählte). Während sich die Entscheidung bei dem ersten beiden Kriterien hauptsächlich aus der Art der Profile ergibt, so können die letzteren genutzt werden, um die Architektur bestimmten Verteilungen von Updates und Nachrichten möglichst gut gerecht zu werden; Genaueres in Abschnitt 4.

3.3 Verknüpfung der Einzelergebnisse

Da die Ergebnisse der einzelnen Indizes im Allgemeinen nicht alle Attribute einer Nachricht abdecken (häufig nur ein einzelnes), müssen die einzelnen Ergebnisse kombiniert werden. Üblicherweise handelt es sich dabei , entsprechend der Profile, um Mengenverknüpfungen wie Vereinigung oder Schnittmenge. Auch hierfür gibt es in der Forschung Konzepte, die diesen Schritt effizienter machen.

3.4 Entfernung von „False positives“

Als letzter Schritt ist die Entfernung von „False positives“ notwendig, falls diese in vorherigen Stufen aufgetreten sind. Dies muss für jedes zu überprüfende Profil separat geschehen, u.U. auch für jeden Wert darin.

4 Existierende Verfahren

Bevor wir den von uns vorgeschlagenen Ansatz vorstellen, sollen kurz existierende Verfahren vorgestellt werden, die eventuell auf das Problem anwendbar sind:

1. **Kein Indexeinsatz:** An erster Stelle ist das Verfahren zu nennen, das auf Indizes komplett verzichtet. Damit sind hohe Raten von Kontext-Updates möglich, da nur eine einfache Zuweisung benötigt wird. Von Nachteil ist allerdings die schlechte Skalierbarkeit für viele Profile und hohe Datenraten, da alle Profile einzeln überprüft werden müssen.
2. **Vollständiges Indexieren der Werte („Full“):** Genau die umgekehrten Eigenschaften hat das bei Informationsfiltern verwendete Verfahren, Indizes auf dem gesamten Wertebereich aller Attribute zu legen und jedes Update sofort umzusetzen. Die gute Skalierbarkeit für viele Profile und hohe Nachrichtenraten wird mit teuren Updates erkauft, da der Index gepflegt werden muss.
3. **Lazy Updates:** Aus dem Bereich der „Moving objects“ kommt der Ansatz, Updates im Index nur dann durchzuführen, falls dies in der Indexstruktur unvermeidlich ist oder der „Sicherheitsbereich“ um ein Objekt überschritten wird[Le03]. Im Gegensatz zum letzten Verfahren ist hier die Entfernung von „false positives“ notwendig.
4. **Partielles Indizieren:** In diesem Verfahren werden Indizes nur bei Attributen mit geringer Updaterate/hoher Selektivität verwendet oder nur in Wertebereichen von Attributen, die ebenfalls solche Eigenschaften haben[ST89]. Mit dieser Technik werden gute Ergebnisse erzielt, falls Verteilungen bekannt und konstant sind. Problematisch ist jedoch die Durchführung durch einen Administrator, da dies aufwendig und fehlerbehaftet ist.

5 Adaptives Indexing

Die im letzten Abschnitt vorgestellten Ansätze haben ihre Vorteile vor allem an den Enden des Spektrums des Verhältnisses Updates zu Nachrichten. Um den mittleren Bereich abzudecken, kombinieren wir Lazy Updates mit einer adaptiven Indexerweiterung, die im Index „false positives“ erlaubt, um Updates zu verbilligen. Den Übergang zwischen den Extremen ermöglichen wir durch neu eingeführte Operationen auf dem Index, `Eskalation` und `De-Eskalation`, die auf einzelne Profile angewandt werden. Eskalation als Schritt in Richtung des Verfahrens ohne Index macht Updates billiger, indem sie einen größeren Bereich von Werten zu einem Profil zuordnet. Als Konsequenz ist jedoch mit mehr „false positives“ zu rechnen. Umgekehrt geht De-Eskalation: Schritt in Richtung „Full Index“, in dem die Menge von Werten, die einem Profil zugeordnet sind, verringert wird: Updates werden teurer, weniger false Positives werden erzeugt.

Entscheidend für die Wirksamkeit dieser Operationen ist, wann diese durchgeführt werden: Eskalation bei jedem Update, bei dem der neu zugewiesene Wert nicht mehr in der Menge der bisher zugeordneten Werte ist. Im Umkehrschluss müssen Updates, deren Werte bereits mit dem Profil assoziiert sind, nicht mehr den Index verändern.

Da ein wiederholtes Anwenden der Eskalation auf Dauer zu einer Degradierung des Index führt, muss sie mit De-Eskalation kompensiert werden. Hierzu verwenden wir das Feedback der false positives, d.h. ein als falsch erkannter Wert in der Eliminationsstufe führt dazu, dass das Profil im Index genauer gemacht wird.

Die hier vorgeschlagenen Operationen sind auf nahezu alle Indextypen anwendbar, allerdings besonders einfach bei hierarchischen Indizes umzusetzen, da hier Eskalation der Übertragung von Identifikatoren zu einem Vorfahren entspricht, während De-Eskalation Identifikatoren in Richtung der Blätter verschiebt. Der Test auf „Enthaltensein“ kann schnell durch Überprüfung der Schlüssel an dem Knoten durchgeführt werden, an dem sich der Identifikator befindet.

6 Experimente

Um die von uns vorgeschlagenen Konzepte zu verifizieren, erweiterten wir eine existierende Implementierung (in C++) von Interval Skip Lists[HJ96] um die neuen Methoden. Die Experimente wurden auf einem Pentium 4 2.8 Ghz unter Linux durchgeführt.

Aus Platzgründen soll hier nur die Ergebnisse des wichtigsten Tests ausführlicher dargestellt werden, weitere Resultate werden nur kurz skizziert.

Da die Leistungsfähigkeit bei verschiedenen Updateraten die Verfahren am deutlichsten unterscheidet, haben diesen Parameter verändert, während die Anzahl der Nachrichten konstant gehalten wurde. Für die Updaterate verwenden wir als Einheit das *Verhältnis von Nachrichten pro Update pro Profil* (MUP), um die Workload von der Anzahl der Profile unabhängig zu machen.

Nachrichten bestanden aus 10 Attributen-Wert-Paaren, ebenso Kontexteinträge. Profile bestehen aus einer Konjunktion von durchschnittlich 7 Vergleichsoperationen, in denen Nachrichtenattribute mit Kontextattributen (80 %) oder Konstanten (20 %) verglichen wurden. Jedes Nachrichtenattribut kommt nur in genau einer Vergleichsoperation vor, wodurch einseitig offene Intervalle entstehen. Insgesamt wurden im hier beschriebenen Experiment 30.000 Profile verwendet. Auf der Basis dieser Daten verglichen für die Verfahren

ohne Index, mit Full Index, Lazy Index und unserem adaptiven Index die Zeit, die benötigt wird um 1000 Nachrichten zu testen, während wir die MUP variierten. Das Ergebnis ist in Abbildung 2 zu sehen:

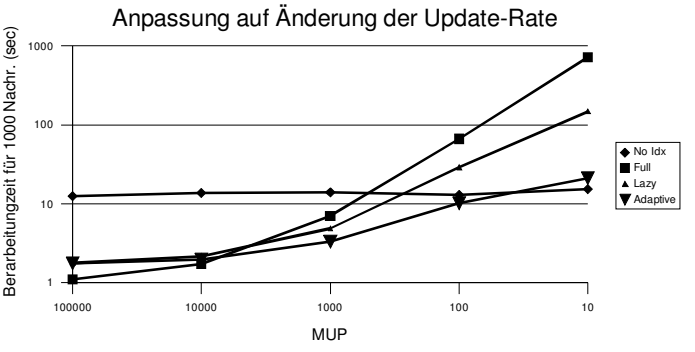

Abbildung 2: Vergleich bei Änderung der Updaterate

Das Verfahren ohne Index wird durch steigende Anzahl von Updates kaum beeinflusst, da es sich dabei nur um Zuweisungen handelt; bei geringer Updaterate ist es aber um eine Größenordnung schlechter als das beste Verfahren. Full Index ist bei geringer Updaterate am besten, jedoch steigen die Kosten bei höheren Updateraten durch die Indexoperationen schnell an; Bei MUP 10 ist es 45-mal schlechter als das beste Verfahren (ohne Index). Lazy erreicht gegenüber Full bei hohen Updateraten bereits eine deutliche Verbesserung, ist dennoch nie das beste Verfahren. Das adaptive Verfahren hat wie bei Lazy bei geringer Updaterate durch Entfernung der „false positives" einen gewissen Overhead, danach steigen die Kosten jedoch nur moderat. Bei MUP 10 ist gegenüber dem Verfahren ohne Index ein leichter Overhead zu sehen. Damit erfüllt das adaptive Verfahren die Erwartungen. Eine Erhöhung der Profilanzahl veränderte die Relation der einzelnen Verfahren nicht maßgeblich, lediglich die absoluten Zeiten stiegen an. Deutlicheren Einfluss hatte die Rate des Feedback von „false positives", da ein sehr starkes Feedback die Performance negativ beeinflusst.

7 Bewertung und Ausblick

Kontextbasierte Profile erweitert Informationsfilter deutlich in Ausdrucksmächtigkeit und Nutzbarkeit. Dabei ergibt sich jedoch das Problem von hohen Updateraten auf den Profilindizes. Unser Vorschlag einer automatischen und schrittweisen Anpassung von bestehenden Indexstrukturen zwischen hoher Nachrichtenrate und hoher Updaterate ermöglicht es, diese Konzepte umzusetzen.

Nächste Schritte auf diesem Gebiet werden die Anwendung auf andere Indexstrukturen, die Integration weiterer Techniken zu Skalierbarkeit sowie verfeinerte Steuerungsmechanismen für bestimmte Aspekt sein.

Literaturverzeichnis:

[De01] A. Dey, Understanding and Using Context, Personal and Ubiquitous Computing Journal 5(1), 2001

[HJ96] E. Hanson and T. Johnson, Selection Predicate Indexing for Active Databases Using Interval Skip Lists, Information Systems 21(3), 1996

[ST89] M. Stonebraker, The Case for Partial Indexes, SIGMOD Record 18(4), 1989

[Le03] M. L. Lee et al. Supporting Frequent Updates in R-Trees: A Bottom-Up Approach, VLDB 2003