Top-N Anfrageverarbeitung in PDMS: Anforderungen und Lösungswege

Katja Hose
katja.hose@tu-ilmenau.de
Fakultät für Informatik und Automatisierung, TU Ilmenau
Postfach 100565, D-98684 Ilmenau

Zusammenfassung

Da P2P-Netzwerke gewöhnlich aus einer großen Anzahl von Peers bestehen und somit große Datenmengen verfügbar sind, ist es oft unnötig wenn nicht sogar unmöglich, dem Nutzer ein vollständiges Ergebnis seiner Anfrage durch Einbeziehung aller teilnehmenden Peers zu präsentieren. Vielmehr muss in P2P-Systemen begründet durch deren Charakteristik zumeist auf best-effort Lösungen zurückgegriffen werden. Ein Beispiel dafür stellen Top-N-Anfragen dar. Als Ergebnis wird ein gemäß der Ähnlichkeit zur gestellten Anfrage sortiertes Ergebnis erwartet, das die „besten“ zum gegebenen Zeitpunkt im Netzwerk vorhandenen Datenwerte widerspiegelt, ohne dabei eine kostenintensive Berechnung vorauszusetzen. Die dabei gestellten Anforderungen an die Anfrageverarbeitung umfassen zumeist eine möglichst geringe Bearbeitungsdauer sowie eine möglichst hohe Korrektheit des Ergebnisses bezüglich aller im Netzwerk vorhandenen Daten. In der vorliegenden Arbeit wird zum Einen vorgestellt, welche Anforderungen an eine Strategie, die Top-N-Anfragen bearbeitet, zu stellen sind. Zum Anderen werden konkrete Lösungsansätze vorgestellt, die die Bearbeitung derartiger Anfragen auf effiziente Art und Weise realisieren.

1 Einführung

Seit einigen Jahren werden schemabasierte Peer-to-Peer (P2P) Systeme im Rahmen aktueller Forschung untersucht. Im Gegensatz zu klassischen Systemen bestehen P2P Systeme, auch Peer Data Management System (PDMS) genannt, aus einer Menge von Peers, welche vollkommen unabhängig voneinander sind. Dies bedeutet unter Anderem, dass es sich um eine dynamische Netzwerkstruktur handelt, in der sich Peers unabhängig voneinander an- und abmelden bzw. ausfallen können. Zudem sind auch lokales Schema sowie die lokal verwalteten Daten selbst keinen Abhängigkeiten zu anderen Peers unterworfen.

Dadurch dass sich PDMS meist durch eine große Anzahl von Peers auszeichnen, ist eine vollständige und exakte Anfrageverarbeitung oftmals nicht möglich, da eine solche Bearbeitung zum Einen das Einbeziehen aller Peers, d.h. das Fluten des Netzwerkes, voraussetzen würde. Zum Anderen stellen unvollständige bzw. inkorrekte Mappings, sowie unvollständige Informationen über Datenverteilung und -lokalisation weitere erschwerende Aspekte dar. Deshalb wird oft auf *best-effort* Lösungen zurückgegriffen, die versuchen, eine Anfrage so exakt und kostenminimal wie möglich zu beantworten, jedoch ein vollständiges Ergebnis nicht garantieren können.

Dies spielt insbesondere bei der Berechnung von Top-N-Anfragen eine wichtige Rolle. Das Ziel dabei ist nicht ein vollständiges und exaktes Ergebnis zu ermitteln, sondern vielmehr *möglichst* alle N besten Ergebnistupel auf effiziente Weise zu finden, wobei natürlich ein Kompromiss zwischen Genauigkeit bzw. Vollständigkeit und Effizienz bzw. Kostenminimierung gefunden werden muss. Die Schwierigkeit besteht vor Allem darin, Kriterien bzw. Regeln zu entwickeln, anhand derer entschieden werden kann, welche Peers nicht in die Anfrageverarbeitung einbezogen werden, ohne dass die Genauigkeit d.h. die Qualität des Ergebnisses zu große Einbußen erfährt. Da in PDMS kein globales Wissen zur Verfügung steht, stellt dies besondere Anforderungen an die verwendeten Anfragebearbeitungsstrategien.

Nach der Vorstellung verwandter Arbeiten in Abschnitt 2 werden in Abschnitt 3 einige grundlegende Ansätze bzw. Strategien der Top-N-Anfrageverarbeitung vorgestellt, ihre Evaluierung folgt in Abschnitt 4. Abschließend umreißt Abschnitt 5 kurz, welche Aspekte und Optimierungen grundlegender Verfahren in aktueller und zukünftiger Forschungsarbeit untersucht werden.

2 Verwandte Arbeiten

Erste Ansätze zur Berechnung von Top-N-Anfragen wurden für die Anwendung in RDBMS konzipiert. Wie in [CG99, BCG02] vorgestellt, erscheint es zunächst sinnvoll, existierende Datenstrukturen wie zum Beispiel Indexe und Statistiken, beispielsweise in Form von Histogrammen, zu verwenden, um eine Top-N-Anfrage in eine traditionelle Select-Anfrage zu übersetzen und diese dann auf übliche Weise zu bearbeiten. Ein anderer Ansatz, der versucht, das Risiko eines Neustarts einer Anfrage auf probabilistische Weise zu quantifizieren, wird in [DR99] vorgestellt.

TPUT [CW04] ist ein Algorithmus, der für die effiziente Berechnung von Top-N-Anfragen auf Aggregaten in verteilten Systemen entwickelt wurde. Dieser Algorithmus stellt eine Weiterentwicklung des *Threshold Algorithmus (TA)* dar, welcher unabhängig von mehreren Gruppen entwickelt wurde [NR99, GBK00, LNF01]. Weitere Varianten wurden entworfen für Multimedia Repositories [CGM04], Distributed Preference Queries auf internetzugänglichen Datenbanken [MBG04] und für das Ranken von Anfrageergebnissen strukturierter Datenbanken [ACDG03]. Eine weitere Arbeit [TWS04] basierend auf TA führt eine Familie von approximativen Top-N Algorithmen ein, die auf probabilistische Art und Weise den Berechnungsaufwand zu minimieren versuchen. All diese Algorithmen benötigen mehrere Round-Trips, um das Endergebnis zu ermitteln, was im allgemeinen Fall als nachteilig angesehen werden muss.

Die in [NSTB04, BNST05] vorgestellten Algorithmen versuchen, die Bearbeitung von Top-N-Anfragen zu verbesseren, indem sie dynamisch Anfragestatistiken sammeln, die beim erneuten Bearbeiten der gleichen Anfrage verwendet werden können, um das Anfragerouting zu verbessern. Wird eine Anfrage ein erstes Mal gestellt, müssen jedoch alle Peers am Prozess der Anfrageverarbeitung teilnehmen, wobei mehrere Round-Trips zum Ermitteln des Endergebnisses benötigt werden.

3 Basisstrategien

Im Gegensatz zu einigen der im vorhergehenden Abschnitt vorgestellten Verfahren basieren die im Folgenden vorgestellten Strategien darauf, dass jeweils nur ein Durchlauf bzw. Round-Trip benötigt wird, um eine Anfrage vollständig zu beantworten. Dies bedeutet, dass jeder Peer eine Anfrage nur ein einziges Mal erhält und niemals mehrfach bezüglich der Bearbeitung einer einzigen Anfrage kontaktiert wird. Dies bringt unter Anderem die folgenden zwei Vorteile mit sich: erstens werden Kosten reduziert, und zweitens werden Probleme vermieden, die sich durch die Tatsache begründen, dass in PDMS aufgrund der Dynamik nicht garantiert werden kann, dass sich das Netzwerk im zweiten Durchlauf noch immer im gleichen Zustand befindet (An- und Abmeldevorgänge, Ausfälle, Updates,...). Zur Vereinfachung, ohne Beschränkung der Allgemeinheit, wird dabei in den folgenden Abschnitten die zugrunde liegende Netzwerkstruktur als Baum aufgefasst. Die vorgestellten Strategien können jeweils durch kleinere Modifikationen zur Zyklenerkennung erweitert werden.

Naiver Ansatz Die einfachste Strategie zur Beantwortung einer Top-N-Anfrage besteht darin, alle Peers in die Anfrageverarbeitung einzubeziehen, d.h. das Netzwerk zu fluten. Dabei bearbeitet jeder Peer die Anfrage lokal und sendet jeweils seine N lokal besten Ergebniselemente zum Initiator. Dieser berechnet nach Erhalt der Antworten das Endergebnis und gibt es als globales Top-N-Ergebnis aus.

Partieller Ansatz Eine Möglichkeit, diesen naiven Ansatz zu verbessern, besteht darin, die Tatsache auszunutzen, dass die Antworten in einem Netzwerk mehrere Peers durchlaufen müssen, bevor sie den Initiator letztendlich erreichen. Auf dem Weg zum Initiator fasst jeder Peer die Daten seiner Nachbarn zusammen, berechnet daraus ein Teilergebnis mit N Elementen und leitet dieses dann weiter. Schließlich verfährt der Initiator auf gleiche Weise und verfügt somit über das Endergebnis, das dem User präsentiert wird. Offensichtlich reduziert diese Strategie das zu sendende Datenvolumen, jedoch nicht die Anzahl involvierter Peers, da das Netzwerk immer noch geflutet werden muss, um alle Peers zu erreichen.

Global optimierender Ansatz Um zusätzlich auch die Anzahl der am Anfrageverarbeitungsprozess teilnehmenden Peers reduzieren zu können, müssen Indexstrukturen existieren, die Auskunft darüber geben, über welche Daten ein Nachbar verfügt bzw. auf welche Daten über einen Nachbarn zugegriffen werden kann. Ist eine Indexstruktur mit globalem Wissen, zum Beispiel auf Basis von DHTs, vorhanden

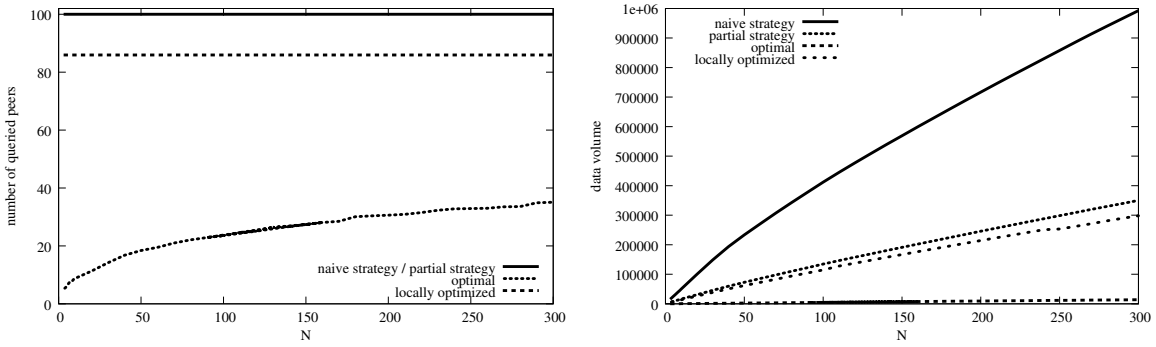
und auf dem angefragten Attribut definiert, so kann der Initiator die Anfrageverarbeitung insbesondere das Routing der Anfrage zentral planen, mittels des vorhandenen globalen Wissens optimieren und somit die Bearbeitungskosten reduzieren.

Lokal optimierender Ansatz Ist hingegen kein globales Wissen vorhanden, so muss jeder Peer, der die Anfrage bearbeitet, die Entscheidung über ein möglichst optimales Routing lokal treffen. Dies geschieht wiederum auf Basis von Indexen, wobei davon ausgegangen wird, dass die Informationen dieser Indexe innerhalb eines bestimmten Horizontes beschränkt sind und somit keine Informationen über Daten enthalten, die in sehr großer Entfernung liegen. Aufgrund der ihm vorliegenden Indexe entscheidet jeder Peer, der die Anfrage erhält, individuell welche seiner Nachbarn über relevante Daten bezüglich der gestellten Anfrage verfügen, leitet die Anfrage an diese Peers weiter und bezieht sie in die Anfrageverarbeitung ein. Dieser im Folgenden als lokal optimierend bezeichnete Ansatz basiert auf dem partiellen Ansatz, um gleichzeitig das Datenvolumen zu reduzieren.

4 Evaluierung

In diesem Abschnitt sollen die grundlegenden Performanceunterschiede der in Abschnitt 3 vorgestellten Basisstrategien aufgezeigt werden. Alle umgesetzten Algorithmen wurden im gleichen statischen Netzwerk bestehend aus 100 Peers ausgewertet. Wie bereits erwähnt, wurde eine Baumstruktur als Netzwerk gewählt, wobei jeder Peer ausgenommen der Blattknoten 4 Kinder besitzt. Als Testdaten wurden Astrodaten [Kha01] verwendet, wobei sich alle Anfragen zur Vereinfachung auf eine Dimension, d.h. ein Attribut beschränken. Ein Teil der Daten wurde geclustert auf die Peers verteilt, so dass jeder Peer viele Daten in einem bzw. auch mehreren Teilbereichen des gesamten Attributwertebereichs besitzt. Ein weiterer Teil wurde zufällig auf alle Peers verteilt. Bezüglich jeder Anfragestrategie wurden zehn Anfragen mit jeweils unterschiedlichen Attributwerten am gleichen Peer initiiert. Die im Folgenden dargestellten Kurven repräsentieren die Mittelung zwischen den Ergebnissen dieser zehn Anfragen, die den gesamten Wertebereich des angefragten Attributes überdecken.

Der Vergleich der Strategien erfolgt auf Basis von Datenvolumen und Anzahl angefragter Peers. Auf die Betrachtung der Anzahl zur Bearbeitung einer Anfrage benötigter Nachrichten wird an dieser Stelle aufgrund der starken Korrelation zur Anzahl involvierter Peers verzichtet. Der Grund für diese Korrelation besteht darin, dass alle umgesetzten Strategien darauf basieren, dass ein Peer auf die Antworten seiner Nachbarn wartet, bevor er eine eigene sendet. Für den lokal optimierenden Ansatz wurde eine Indexstruktur auf Basis von Histogrammen verwendet, deren Horizont stets alle Peers beinhaltet. (In der gewählten Netzwerkstruktur beträgt die maximale Entfernung eines beliebigen Peers vom Initiator 4)



(a) Anzahl involvierter Peers

(b) Benötigtes Datenvolumen

Abbildung 1: Betrachtungen in statischen Netzwerken

Abbildung 1 zeigt das grundsätzliche Verhalten von naivem, partiellem und lokal optimierendem Ansatz in statischen Netzen. Die als „optimal“ bezeichneten Kurven sollen lediglich zu Vergleichszwecken dienen. Sie stellen minimale Werte dar, die nur dann erreicht werden können, wenn (i) eine Anfrage auf optimalem Weg nur zu genau denjenigen Peers geleitet wird, die tatsächlich über die N besten Datensätze im ganzen Netzwerk verfügen, und (ii) diese Peers auch nur genau diejenigen Elemente zurückliefern,

die zum exakten Endergebnis gehören. Der global optimierende Ansatz wurde nicht weiter verfolgt, da der Fokus unserer Arbeit auf PDMS und den einhergehenden Problemstellungen der Dynamik und begrenztem Wissen liegt.

Abbildung 1(a) zeigt, wie sich die Anzahl der an der Anfrageverarbeitung teilnehmenden Peers bei steigendem N verhält: Bei naiver und partieller Strategie werden stets alle 100 Peers des Netzwerkes angefragt. Bei der lokal optimierenden Strategie fällt auf, dass immer gleichbleibend viele Peers involviert werden. Dies ist vor Allem dadurch begründet, dass stets alle Peers einbezogen werden müssen, die über Daten verfügen, die *eventuell* zu den N global besten gehören, d.h. Datensätze mit Attributwerten nahe dem angefragten Wert. Begründet durch die Datenverteilung zeigen sich bei veränderlichem N keine Unterschiede.

Das benötigte Datenvolumen, welches in Abbildung 1(b) dargestellt ist, bestätigt die Erwartung, dass die lokal optimierende Strategie weniger Datenvolumen benötigt als der partielle Ansatz. Dies lässt sich anhand der Tatsache begründen, dass die lokal optimierende Strategie weniger Peers in den Anfrageprozess involviert, siehe Abbildung 1(a), und somit auch weniger Datenvolumen zur Anfrageverarbeitung nötig ist. Im Gegensatz zur naiven Strategie, welche alle Daten zunächst am Initiator sammelt und deshalb das größte Datenvolumen verursacht, versendet jeder Peer bei Anwendung des lokal optimierenden oder partiellen Ansatzes maximal N Elemente als Antwort.

Zusammenfassend lässt sich sagen, dass der lokal optimierende Ansatz im Vergleich zur naiven bzw. partiellen Strategie weniger Kosten (Datenvolumen bzw. Peeranzahl) verursacht. Wie in weiteren hier aus Platzgründen nicht aufgeführten Tests gezeigt werden konnte, hat weder Datenverteilung noch Netzwerkstruktur Einfluss auf diese grundsätzliche Feststellung. Die lokal optimierende Strategie arbeitet in statischen Netzwerken derart, dass stets das exakte globale Top-N-Ergebnis zurückgeliefert wird. Dabei werden oftmals viele Peers angefragt, die zwar dem angefragten Schema entsprechende Daten besitzen, letztendlich jedoch nichts zum Endergebnis beitragen können. Der Grund dafür ist die Tatsache, dass Indexe stets eine Approximation der Realität darstellen und der Algorithmus dies durch das *zusätzliche* Anfragen weiterer Peers ausgleicht und so Fehler vermeidet. Folglich muss eine Strategie, deren Kosten niedriger als die der „lokal optimierenden“ Strategie sein sollen, d.h. sich denen als „optimal“ bezeichneten Kosten annäheren, „bewusst“ das Risiko eingehen, dass das Ergebnis nicht alle N besten Elemente des Netzwerkes widerspiegelt. Nur dann können Kostenreduktionen durch das Nichtanfragen von Peers erzielt werden. Dies stellt neben weiteren im nächsten Abschnitt vorgestellten Aspekten einen Schwerpunkt unserer momentanen Forschungsarbeit dar.

5 Ausblick

Wie in den vorhergehenden Abschnitten bereits aufgezeigt, ist es das Ziel weiterer Forschung, best-effort Lösungen zu entwickeln, die es erlauben, die Anzahl involvierter Peers zu reduzieren. Eine Möglichkeit, den dabei eingegangenen Fehler zu quantifizieren, stellen Wahrscheinlichkeitsgarantien dar. Zum Beispiel könnte ein Nutzer beim Stellen einer Anfrage eine Fehlergrenze angeben, die es dem zugrunde liegenden Algorithmus ermöglicht, Fehler einzugehen. Ein Anfrageergebnis könnte somit in folgender Form angegeben werden: „mit einer Wahrscheinlichkeit von 90% entspricht das ermittelte Ergebnis dem global exakten“.

Im Gegensatz zu den hier untersuchten Netzwerken zeichnen sich reale P2P-Systeme meist durch ein hohes Maß an Dynamik aus. Dieser wichtige Aspekt stellt besondere Anforderungen an Algorithmen, die in solchen Netzwerken eingesetzt werden sollen. Durch eine inkrementelle Arbeitsweise, die prinzipiell bei allen vorgestellten Basisstrategien realisiert werden kann, wird zwar u.U. Datenvolumen und Nachrichtenanzahl erhöht, jedoch sind diese Varianten robuster gegenüber dynamischen Veränderungen der Netzwerkstruktur [KHS04]. Im Vergleich der einzelnen inkrementellen Strategievarianten untereinander würde sich jedoch das in Abschnitt 4 aufgezeigte Kostenverhältnis nicht ändern.

Insbesondere die Tatsache, dass Indexe zumeist nur auf einen gewissen Horizont beschränkt sind, stellt einen interessanten in zukünftiger Arbeit zu untersuchenden Aspekt dar. Gleiches gilt für die Untersuchung der Korrektheit des Ergebnisses unter Einwirkung von verschiedenen Indexaktualisierungsstrategien (Query-Feedback, epidemische Protokolle, Gossiping...).

Allgemeine Einsparungen von Datenvolumen könnten durch das Mitsenden von lokalen Ergebnisinformationen erzielt werden. Dadurch dass jeder Peer, der eine Anfrage weiterleitet, Informationen darüber mitsendet, welche Ergebniswerte ihm bereits bekannt sind, können die Empfänger-Peers ihre Antworten dahingehend optimieren, dass sie nur solche Datenelemente enthalten, die *besser* sind als die bereits bekannten.

Aus Platzgründen können an dieser Stelle nicht alle Aspekte ausführlich erläutert werden. Zusammenfassend lässt sich sagen, dass sich unsere momentane Forschungsarbeit damit beschäftigt, eine Anfragestrategie zu entwickeln, welche die folgenden Charakteristika erfüllt: inkrementelle Arbeitsweise, Beantwortung einer Anfrage innerhalb eines Round-Trips, Nutzung vorhandener Indexstrukturen, Kostenminimierung unter Angabe von probabilistischen Garantien für die Vollständigkeit bzw. Korrektheit des Ergebnisses.

Literatur

[ACDG03] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, and Aristides Gionis. Automated ranking of database query results. In *CIDR*, 2003.

[BCG02] N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM TODS, Vol. 27, No. 2*, 2002.

[BNST05] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top k retrieval in peer-to-peer networks. In *ICDE'05*, 2005.

[CG99] Surajit Chaudhuri and Luis Gravano. Evaluating top-k selection queries. In *VLDB'99*, pages 397–410, 1999.

[CGM04] Surajit Chaudhuri, Luis Gravano, and Amélie Marian. Optimizing queries over multimedia repositories. *IEEE TKDE*, 16(8):992–1009, 2004.

[CW04] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In *PODC '04*, pages 206–215, 2004.

[DR99] Donko Donjerkovic and Raghu Ramakrishnan. Probabilistic optimization of top n queries. In *VLDB'99*, pages 411–422, 1999.

[GBK00] Ulrich Güntzer, Wolf-Tilo Balke, and W. Kiessling. Optimizing multi-feature queries for image databases. In *VLDB 2000*, pages 419–428, 2000.

[Kha01] N. V. Kharchenko. All-sky compiled catalogue of 2.5 million stars (ascc-2.5), 2001. Kinematics and Physics of Celestial Bodies, 17, N 5, 409 - 423; ftp://ftp.mao.kiev.ua/pub/astro/cc.

[KHS04] M. Karnstedt, K. Hose, and K.-U. Sattler. Query Routing and Processing in Schema-Based P2P Systems. In *DEXA'04 Workshops*, pages 544–548. IEEE Computer Society, 2004.

[LNF01] Amnon Lotem, Moni Naor, and Ronald Fagin. Optimal aggregation algorithms for middleware. In *PODS'01*, March 03 2001.

[MBG04] Amélie Marian, Nicolas Bruno, and Luis Gravano. Evaluating top- queries over web-accessible databases. *ACM TODS'04*, 29(2):319–362, 2004.

[NR99] S. Nepal and M. V. Ramakrishna. Query processing issues in image (multimedia) databases. In *ICDE '99*, page 22, 1999.

[NSTB04] Wolfgang Nejdl, Wolf Siberski, Uwe Thaden, and Wolf-Tilo Balke. Top-k Query Evaluation for Schema-Based Peer-to-Peer Networks. In *Int. Semantic Web Conf.*, pages 137–151, 2004.

[TWS04] Martin Theobald, Gerhard Weikum, and Ralf Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB 2004*, pages 648–659, 2004.