Synergie-basiertes Scheduling von Datenbank-Workloads

Martina-Cezara Albutiu, Andreas Scholz, Stefan Krompass, Alfons Kemper

Technische Universität München, Garching, Deutschland
{albutiu,scholza,krompass,kemper}@in.tum.de

Zusammenfassung

Moderne Datenbanksysteme werden mit Workloads bestehend aus Anfragen mit sehr unterschiedlichen Charakteristika konfrontiert. Durch die parallele Ausführung der Anfragen lässt sich der Durchsatz eines Datenbanksystems deutlich steigern, da Systemressourcen besser ausgelastet werden. Der erreichbare Leistungsgewinn hängt neben der verwendeten Hardware auch von den Eigenschaften der betrachteten Anfragen ab. Manche Anfragekombinationen besitzen besonders große Synergien, z.B. durch die Nutzung gleicher, im Datenbankpuffer vorgehaltener Daten oder durch einen komplementären Ressourcenbedarf. Andere Kombinationen hingegen behindern sich gegenseitig bei der Ausführung, was im schlimmsten Fall sogar zu einer Reduzierung des Durchsatzes im Vergleich zur sequentiellen Ausführung der Anfragen führen kann.

Durch intelligentes Scheduling von Datenbankanfragen kann ein deutlicher Leistungsgewinn bei der Anfragebearbeitung in der Datenbank erreicht werden. In diesem Paper wird ein Monitoring-basierter Ansatz entwickelt, der im laufenden Betrieb eine vollautomatische Erkennung und Ausnutzung von Anfragesynergien erlaubt.

1 Einleitung

Um verfügbare Systemressourcen möglichst gut auszunutzen, werden in modernen Datenbanksystemen mehrere Anfragen gleichzeitig bearbeitet. In der Praxis treten häufig Wechselwirkungen zwischen den laufenden Anfragen auf. Diese können entweder positiven Einfluss auf die Verarbeitungsgeschwindigkeit der Anfragen haben, z.B. durch die Verwendung im Puffer vorgehaltener Daten, oder die Gesamtleistung des Datenbanksystems verschlechtern, z.B. beim konkurrierenden Zugriff auf gemeinsam genutzte Ressourcen wie CPU oder Hauptspeicher.

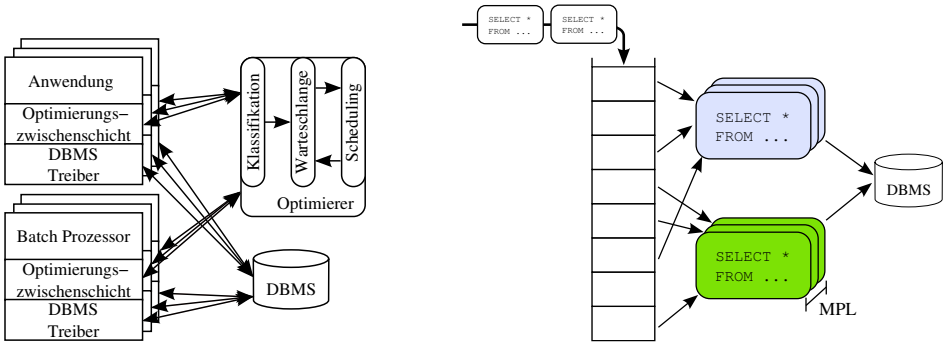
Durch Scheduling, d.h. die gezielte Festlegung welche Anfragen parallel und in welcher Reihenfolge verarbeitet werden, können die positiven Einflüsse zwischen Anfragen zur Durchsatzsteigerung ausgenutzt werden. Gleichzeitig kann sichergestellt werden, dass keine unerwarteten Verzögerungen durch sich gegenseitig behindernde Anfragen auftreten. Grundvoraussetzung für Optimierungen dieser Art ist, die Wechselwirkungen zwischen Anfragen zu erkennen. Im Allgemeinen ist die Vorhersage von Wechselwirkungen schwierig, weil eine Fülle von Faktoren zu ihrem Entstehen beitragen kann, z.B. die verwendete Hardware, Konfigurationsparameter, Implementierungsdetails der einzelnen Datenbankoperatoren oder andere Programme, die auf dem Datenbankrechner ausgeführt werden. Zuverlässige Vorhersagemodelle sind daher sehr komplex und oft darauf angewiesen, viele (z.T. unsichere) Annahmen zu treffen. Wir verfolgen im Gegensatz dazu einen Monitoring-basierten Ansatz, welcher die Datenbank als Black-Box betrachtet. Durch die kontinuierliche Überwachung der Antwortzeiten von Datenbankanfragen können die positiven und negativen Wechselwirkungen zwischen Anfragen im laufenden Betrieb ermittelt und für das Scheduling verwendet werden.

2 Verwandte Arbeiten

Einige Workload Management-Techniken sind bereits in Produkten wie dem HP Workload Manager für Neoview [HP, 2007], dem IBM Query Patroller für DB2 [Niu et al., 2006], dem Microsoft SQL Server [Microsoft, 2007] und dem Oracle Database Resource Manager [Oracle, 2001] implementiert. In [Krompass et al., 2008] findet sich eine Übersicht über aktuelle Workload Management-Techniken und -Implementierungen. Üblicherweise werden Ansätze wie FIFO oder prioritätsbasiertes Scheduling verwendet. Allerdings berücksichtigt keines der Produkte Interaktionen zwischen Anfragen.

Die Ansätze für das effiziente Scheduling von Datenbank-Workloads unter Berücksichtigung von Wechselwirkungen zwischen Anfragen lassen sich nach der Art der Informationsgewinnung in Analyse- und Monitoring-Verfahren einteilen: *Analyse-Verfahren* untersuchen die Struktur der Anfragen eines Workloads, um mögliche Synergiequellen wie z.B. ähnliche Unteranfragen [Choenni et al., 1997, Subramanian and Venkataraman, 1998] oder gemeinsam benutzte Tabellen zu erkennen. Beispielsweise untersuchen Zukowski et al. [Zukowski et al., 2007] kooperative Scans als Methode zur Performancesteigerung durch gemeinsam genutzte Daten. Ein Kernproblem der Analyseverfahren ist, dass die zu analysierenden Wechselwirkungen von vielen unterschiedlichen Faktoren abhängen, welche oft auf Grund unzureichender Informationen nur durch Schätzungen angenähert werden können. Ein Beispiel dafür ist die Bestimmung des Inhalts des Datenbankpuffers, welcher außerhalb der Datenbank nur schwer erfasst werden kann.

Monitoring-Verfahren hingegen betrachten die Datenbank und die bearbeiteten Anfragen als Black-Box, über die a priori keine Informationen vorliegen. Durch Beobachtung des Bearbeitungsverhaltens von Anfragen wird Wissen gewonnen, das



(a) Architektur (b) Scheduling

Abbildung 1: Optimierer

auf potenzielle Synergien schließen lässt. O'Gorman et al. [O'Gorman et al., 2002] wenden einen Monitoring-Ansatz an und vergleichen die Anzahl der Plattenzugriffe für die paarweise parallele und sequenzielle Ausführung der Anfragen des TPC-H Benchmarks. Diese Vorgehensweise lässt jedoch nur Rückschlüsse auf Wechselwirkungen zwischen genau zwei Anfragen zu und setzt zudem voraus, dass dem Scheduling eine Phase vorausgeht, in der alle Anfragenpaare durchgeführt werden. Auch Krompass et al. [Krompass et al., 2007] verfolgen in ihrem Framework einen Monitoring-Ansatz für effizientes Workload Management, der u.a. aus der Bearbeitungszeit von Anfragen in der Datenbank Scheduling-Entscheidungen ableiten kann. Das Framework geht jedoch von keiner konkreten Scheduling-Implementierung aus.

Wir stellen in diesem Paper einen Monitoring-basierten Ansatz zur Workload-Optimierung vor, der die bereits in [Krompass et al., 2008] vorgestellten Konzepte aufgreift. Unabhängig von der hier vorgestellten Lösung verfolgen Ahmad et al. [Ahmad et al., 2008] einen ähnlichen Ansatz. Wir ermitteln basierend auf der beobachten Ausführungszeit Synergien zwischen den enthaltenen Anfragen und zeigen wie diese mit Hilfe von Scheduling-Algorithmen zur effizienten Bearbeitung von Datenbank-Workloads genutzt werden können.

3 Monitoring-basierte Workload-Optimierung

Das Anfrage-Scheduling nutzt als Eingabe eine Menge von Anfragen, die im Datenbanksystem ausgeführt werden sollen. Diese Anfragen werden dem System entweder als Batch zur Verfügung gestellt oder interaktiv von Benutzern abgesetzt. Um eine Überlastung der Datenbank zu vermeiden, werden die Anfragen vor ihrer Ausführung abgefangen und in einer Warteschlange vorgehalten. Der Scheduler ermittelt aus der Menge der wartenden Anfragen die Teilmenge, die als nächstes ausgeführt werden soll. Die Auswahl wird dabei so getroffen, dass der Durchsatz des Datenbanksystems maximiert wird. Eine beispielhafte Architektur für ein DBMS mit Workload-Optimierung wird in den folgenden Abschnitten vorgestellt.

3.1 Architektur

Abbildung 1(a) zeigt eine datenbankunabhängige Referenzarchitektur für ein DBMS mit Workload Management-Funktionalität. Eine Optimierungszwischenschicht fängt auf Client-Seite an die Datenbank gerichtete Anfragen ab und leitet sie zum Workload-Optimierer um. Diese Zwischenschicht kann durch einen Wrapper um den Datenbanktreiber implementiert werden, z.B. einen modifizierten JDBC-Treiber für Java-Clients. Batches werden ebenfalls zunächst an den Workload-Optimierer gesendet. Im Optimierer werden eintreffende Anfragen klassifiziert, d.h. einem Anfragetypen zugeordnet. Grund hierfür ist, dass sich viele Anfragen durch eingesetzte Parameterwerte unterscheiden, ihr Ausführungsverhalten hingegen jedoch identisch ist. Der Optimierer arbeitet deshalb mit Anfragetypen, welche durch die Entfernung der Konstanten aus der Anfrage extrahiert werden können. Die klassifizierten Anfragen werden anschließend in eine Warteschlange einsortiert. Ein Scheduling-Algorithmus ermittelt aus der Menge der wartenden Anfragen die Teilmenge, die als nächstes in der Datenbank bearbeitet werden soll. Bei der Wahl der Anfragen wird darauf geachtet, dass die Anzahl der gleichzeitig ausgeführten Anfragen, der Multi-Programming-Level (MPL), so gewählt wird, dass der Durchsatz in der Datenbank maximal ist. Wird eine Anfrage zur Bearbeitung in der Datenbank freigegeben, wird dies an die Client-seitige Zwischenschicht bzw. den Batch-Prozessor gemeldet, welche wiederum die Ausführung der Anfragen an den Datenbanktreiber delegieren. Alternativ zur vorgestellten Architektur kann der Optimierer auch direkt als Bestandteil des DBMS implementiert werden, die Client-seitige Zwischenschicht kann in diesem Fall entfallen. Um eine datenbankunabhängige Lösung zu erhalten, wurde für diese Arbeit die erste Variante gewählt.

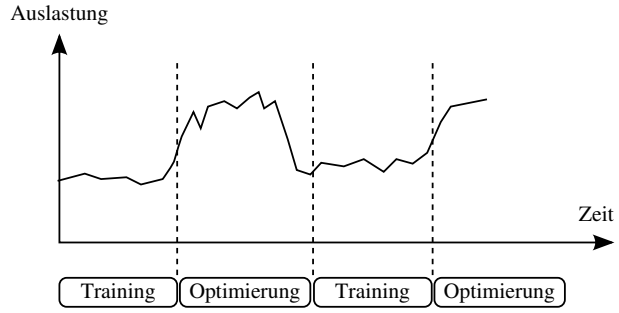


Abbildung 2: Scheduling-Phasen

3.2 Phasen der Workload-Optimierung

Die Workload-Optimierung kann in zwei Phasen unterteilt werden: die initiale *Trainingsphase* und die anschließende *Optimierungsphase*. Während das Scheduling in der Trainingsphase das Ziel verfolgt, die positiven und negativen Einflüsse der Anfragetypen zu ermitteln, liegt der Fokus des Schedulings in der Optimierungsphase auf der Durchsatzmaximierung durch die Anwendung des gewonnenen Wissens.

Zu Beginn der Trainingsphase sind keine Wechselwirkungen zwischen Anfragetypen bekannt. Durch die Ausführung der Anfragen in bestimmten Kombinationen wird versucht, innerhalb kurzer Zeit ein möglichst breites Wissen über das Verhalten der Anfragen zu erhalten. In dieser Phase wird nicht die Optimierung der Datenbankleistung verfolgt, sondern ein maximaler Wissenserwerb. Um nicht Gefahr zu laufen andere, noch größere Wechselwirkungen zu übersehen, wird in der Trainingsphase vermieden, bereits gefundene Wechselwirkungen gezielt für das Scheduling zu verwenden.

In der Optimierungsphase wird versucht einen möglichst hohen Durchsatz bei der Ausführung der Anfragen in der Datenbank zu erreichen. D.h. es werden die Anfragen mit den höchsten positiven Wechselwirkungen zusammen ausgeführt. Auch in der Optimierungsphase werden die beobachteten Anfragezeiten über eine Feedback-Schleife zur Wissensbasis hinzugefügt, so dass sie im Weiteren zur Erkennung von Wechselwirkungen herangezogen werden können. Sollte es ohne zu große Leistungsverluste möglich sein, werden in der Optimierungsphase neue Anfragekombinationen auf Wechselwirkungen hin getestet. Diese Vorgehensweise stellt sicher, dass die eventuell noch unzureichenden Informationen nach der Trainingsphase im laufenden Betrieb vervollständigt werden. Außerdem können dadurch langfristige Änderungen bei den Wechselwirkungen erkannt und neu hinzukommende Anfragetypen in die Optimierung integriert werden.

Der optimale Zeitpunkt des Wechsels von der Trainings- zur Optimierungsphase hängt im Allgemeinen vom Datenbank-Workload ab. Während der Fokus in der Trainingsphase auf der Erkundung unbekannter Anfragekombinationen liegt, wird in der Optimierungsphase vornehmlich das bereits gewonnene Wissen ausgeschöpt. In beiden Phasen wird die Wissensbasis über das Bearbeitungsverhalten von Anfragekombinationen erweitert, weshalb der exakte Zeitpunkt des Wechsels nicht entscheidend ist. Das System ist nach der Trainingsphase nicht auf das bis dahin gesammelte (womöglich unvollständige) Wissen beschränkt und kann sich dynamisch an veränderte Bedingungen anpassen. Zusätzlich kann bei variabler Auslastung des Datenbanksystems vom Optimierer adaptiv zwischen den Phasen gewechselt werden, wie in Abbildung 2 dargestellt. Fällt die Auslastung unter ein kritisches Maß, werden die frei gewordenen Ressourcen durch Umschalten auf die Trainingsphase zur zielgerichteten Bestimmung unbekannter oder ungenauer Anfragesynergien verwendet. In Phasen hoher Auslastung wird die optimale Systemleistung durch Verwendung der ermittelten Synergien in der Optimierungsphase gewährleistet. Die Algorithmen, die in den beiden Phasen zum Einsatz kommen, werden in den folgenden Abschnitten erläutert.

4 Synergie-basierte Workload-Optimierung

Wir gehen davon aus, dass der MPL, also die Anzahl an parallel bearbeiteten Anfragen in der Datenbank, fest ist und so gewählt wurde, dass der Datenbankdurchsatz optimal ist. Da heutige Datenbanksysteme keine detaillierten Informationen über die jeweils aktuelle Ressourcenauslastung zur Verfügung stellen, gehen wir im Folgenden davon aus, dass ein fester MPL auf Basis von Expertenwissen bestimmt wird. Eine mögliche Erweiterung besteht in der dynamischen Anpassung des MPLs zur Laufzeit (wie in [Schroeder et al., 2006] beschrieben). Die zweite Annahme ist, dass die Anfragen blockweise ausgeführt werden, d.h. ein Block aus MPL-vielen Anfragen wird erst gestartet wenn der vorherige Block beendet wurde. Dies erleichtert die Erkennung von Wechselwirkungen, da Anfragen immer zum gleichen Zeitpunkt mit der Ausführung beginnen.

Die Synergie zwischen einem Satz von MPL-vielen Anfragen ist definiert als die Laufzeitersparnis (oder -verlängerung), die durch die parallele Ausführung dieser Anfragen auftritt im Vergleich zur durchschnittlichen Laufzeit, die diese Anfragen in allen möglichen Kombinationen benötigen. Die Synergie zwischen zwei Anfragen ist damit kein absoluter Wert, sondern hängt davon ab, welche anderen Anfragen im Workload vorhanden sind. Durch die relative Definition der Synergie ist es nicht notwendig,

die Anfragen zunächst in einer neutralen Umgebung zu testen. Stattdessen können Wechselwirkungen direkt im laufenden Betrieb ermittelt werden. Der vorgestellte Ansatz basiert auf der Annahme, dass sich die beobachtete Bearbeitungszeit (response time, rt) für einen Satz von MPL-vielen Anfragen als Linearkombination von $\binom{MPL}{2}$ paarweisen Einflussfaktoren darstellen lässt. Diese Einflussfaktoren lassen sich als Anteile (shares, s) an der Gesamtlaufzeit des Satzes von Anfragen darstellen. Die Bearbeitungszeit von vier parallel ausgeführten Anfragen a, b, c und d lässt sich demnach folgendermaßen darstellen:

$$rt_{abcd} = s_{ab} + s_{ac} + s_{ad} + s_{bc} + s_{bd} + s_{cd}$$

Jeder Anteil s_{xy} repräsentiert den Einfluss, den die simultane Ausführung der Anfragen x und y auf die Gesamtlaufzeit hat. Haben alle Anfragen gleich lange Bearbeitungszeiten und besitzen auch keine Wechselwirkungen, so ist der Einfluss gleichverteilt auf alle einzelnen Anteile. Bestehen Synergien zwischen zwei Anfragen, ist der Wert des entsprechenden Anteils und damit auch die Gesamtlaufzeit kleiner. Besitzen Anfragen unterschiedliche Komplexität und damit unterschiedliche Bearbeitungsdauer in der Datenbank, wird dies durch einen entsprechend hohen Wert aller Anteile mit den komplexen Anfragen und entsprechen niedrigen Wert der Anteile mit den einfachen Anfragen abgebildet. Die individuellen Anteile können in einer zweidimensionalen Matrix gespeichert werden.

Die Synergie syn_{xy}, d.h. der Laufzeiteinfluss auf x auf Grund der gleichzeitigen Ausführung von y, wird berechnet als die Differenz des durchschnittlichen Laufzeitanteils der Ausführung von x mit jedem der Anfragetypen q im Workload und des Laufzeitanteils der Kombination aus x und y, formal:

$$syn_{xy} = \frac{\sum_{q \in Q} s_{xq}}{|Q|} - s_{xy},$$

wobei Q die Menge aller dem System (bisher) bekannten Anfragetypen darstellt. Positive Synergiewerte repräsentieren die Laufzeitersparnis für x welche entsteht weil y simultan ausgeführt wird, negative Werte eine entsprechende Laufzeitverlängerung.

Wie die Einflussfaktoren, können auch die Synergien in einer Matrix, der *Synergiematrix*, gespeichert werden. Im Gegensatz zu den Laufzeitanteilen ist die Synergie nicht symmetrisch definiert. Es kann der Fall auftreten, dass eine der beiden Anfragen mehr als die andere Anfrage von der gleichzeitigen Ausführung profitiert, z.B. wenn die Anfragen sehr unterschiedliche Laufzeiten haben. Entscheidend ist, dass jeder der Anteile s_{xy} nicht nur in einer Gleichung auftritt, sondern bei allen Anfragekombinationen die x und y enthalten. Durch die Messung der Gesamtlaufzeit verschiedener Anfragekombinationen lassen sich die einzelnen Einflussfaktoren zunehmend genauer quantifizieren. Dazu werden alle Messungen in ein lineares Gleichungssystem bestehend aus Gleichungen der Form (hier beispielhaft für einen MPL von 4)

$$rt_{abcd} = s_{ab} + s_{ac} + s_{ad} + s_{bc} + s_{bd} + s_{cd} + e_{abcd}$$

eingetragen. Die Fehlervariable e_{abcd} dient dazu, Messungenauigkeiten und externe Einflüsse aufzufangen. Wird eine Kombination mehrfach in der Datenbank ausgeführt, wird die linke Seite der Gleichung durch den Mittelwert der beobachteten Ausführungszeiten ersetzt. Eine erneute Ausführung einer bereits bekannten Kombination erlaubt es damit, zufällige Abweichungen durch äußere Einflüsse zu kompensieren. Die Ausführung einer neuen Kombination hingegen ermöglicht die genauere Bestimmung der einzelnen Einflussfaktoren, weil mehr Information bei der Lösung des Gleichungssystems zur Verfügung steht.

Da das Gleichungssystem im Allgemeinen unterbestimmt ist, d.h. weniger Gleichungen als Kombinationen enthalten sind, muss bei der Lösung auf Optimierungsalgorithmen zurückgegriffen werden. Dabei wird das Prinzip der Entropiemaximierung verfolgt, das bereits von Markl et al. [Markl et al., 2007] auf die Abschätzung von Selektivitäten für zusammengesetzte Prädikate angewendet wurde. Dem Prinzip der Entropiemaximierung zufolge wird das gesamte vorhandene Wissen zur Bestimmung der Einflussfaktoren verwendet, während gleichzeitig keine weiteren Annahmen über die Verteilung der Einflussfaktoren getroffen werden. Für die durch fehlendes Wissen gegebene Unsicherheit wird Gleichverteilung angenommen, d.h. es wird die allgemeinste Lösung gewählt, die konform mit der bisher bekannten Information ist.

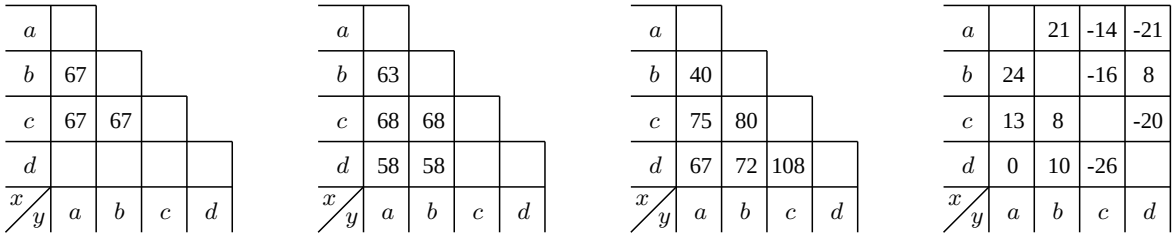
Das erste Ziel der Optimierung ist deshalb, den Einfluss der einzelnen Faktoren auf die beobachtete Ausführungszeit möglichst gleich zu verteilen - sofern keine anderweitigen Erkenntnisse durch Laufzeitmessungen vorliegen. Das zweite Ziel der Optimierung ist, die entstehenden Fehlerkoeffizienten möglichst gering zu halten. Die Zielfunktion der Optimierung ist damit gegeben mit:

$$opt = min \left\{ \frac{\sigma(S)}{avg(S)} + \sum_{e \in E} e \right\},$$

wobei σ die Standardabweichung, avg der Mittelwert, E die Menge aller Fehlerkoeffizienten e und S die Menge aller Anteile s_{xy} ist. Durch den Variationskoeffizienten $\frac{\sigma(S)}{avg(S)}$ wird eine Gleichverteilung der Einflüsse auf die einzelnen Einflussfaktoren erreicht. Die Fehlersumme $\sum_{e \in E} e$ stellt sicher, dass die entstehenden Fehlerterme minimiert werden.

Wir wollen den oben beschriebenen Ansatz an einem Beispiel veranschaulichen. Dazu nehmen wir an, dass ein MPL von 3 festgelegt ist und 4 verschiedene Anfragetypen a, b, c und d (bisher) bekannt sind. Zunächst wird die Kombination bestehend aus den Typen a, b und c ausgeführt, die gemessene Ausführungszeit beträgt 200. Daraus ergibt sich die erste Gleichung:

$$rt_{abc} = s_{ab} + s_{ac} + s_{bc} + e_{abc} = 200$$



x / y	a	b	c	d
a				
b	67			
c	67	67		
d				

(a) s_{xy} bei einer Gleichung

x / y	a	b	c	d
a				
b	63			
c	68	68		
d	58	58		

(b) s_{xy} bei zwei Gleichungen

x / y	a	b	c	d
a				
b	40			
c	75	80		
d	67	72	108	

(c) s_{xy} bei vier Gleichungen

x / y	a	b	c	d
a		21	-14	-21
b	24		-16	8
c	13	8		-20
d	0	10	-26	

(d) Synergiewerte syn_{xy}

Abbildung 3: Entwicklung der Synergiematrix

Nach Lösung des (noch sehr dünn besetzten) Gleichungssystems, ergibt sich für die Variablen s_{ab}, s_{ac} und s_{bc} auf Grund der Gleichverteilung der Unsicherheit der Wert 67, wie in Abbildung 3(a) gezeigt. Für die übrigen Variablen sind noch keine Werte bekannt (dargestellt durch leere Felder in der Matrix). Als nächstes werden die Anfragetypen a, b und d parallel ausgeführt. Diese Kombination wird in 180 bearbeitet. Es wird eine neue Gleichung zum System hinzugefügt:

$$rt_{abd} = s_{ab} + s_{ad} + s_{bd} + e_{abd} = 180$$

Die Lösung des Gleichungssystems ergibt folgende Einflussfaktoren, die in Abbildung 3(b) dargestellt sind: $s_{ab} = 63$, $s_{ac} = 68$, $s_{bc} = 68$, $s_{ad} = 58$ und $s_{bd} = 58$.

Es ist bereits erkennbar, dass die Anteile s_{ad}, s_{bd} und s_{ab} im Vergleich zu s_{ac} und s_{bc} mit kleineren Werten zur Gesamtlaufzeit beitragen und dass die korrespondierenden Anfragepaare somit „synergetischer“ sind. Nach Ausführung einer dritten Kombination a, c und d und einer vierten Kombination b, c und d wird das Gleichungssystem um folgende Gleichungen ergänzt:

$$rt_{acd} = s_{ac} + s_{ad} + s_{cd} + e_{acd} = 250$$

$$rt_{bcd} = s_{bc} + s_{bd} + s_{cd} + e_{bcd} = 260$$

Außerdem wird die Kombination a, b und c ein zweites Mal ausgeführt, diesmal mit einer Ausführungszeit von 190. Die im Gleichungssystem gespeicherten Informationen passen sich an auftretende Änderungen an, indem der Mittelwert der Ausführungszeiten gebildet wird, in diesem Fall 195. Somit wird die erste Gleichung folgendermaßen aktualisiert:

$$rt_{abc} = s_{ab} + s_{ac} + s_{bc} + e_{abc} = 195$$

Die entstehenden Einflussfaktoren sind in Abbildung 3(c) zu sehen. Die aus diesen Werten berechnete Synergiematrix ist in Abbildung 3(d) dargestellt. Wie die relativ kurzen Laufzeiten der Kombinationen mit a und b bereits vermuten lassen, besteht zwischen diesen beiden Anfragetypen eine hohe Synergie. Sowohl a als auch b profitieren durch die parallele Ausführung der jeweils anderen Anfrage, beide Einträge in der Synergiematrix sind positiv: $syn_{ab} = 21$ und $syn_{ba} = 24$. Wie oben beschrieben, stellen die Synergien einen relativen Wert dar, der das Laufzeitverhalten einer Anfrage x in Kombination mit einer zweiten Anfrage y im Vergleich zu dem Laufzeitverhalten von x mit allen bekannten Anfragetypen abbildet. Der Einfluss, den x auf y ausübt, ist nicht notwendigerweise der gleiche, den y auf x ausführt. Im Beispiel profitiert c von der gleichzeitigen Ausführung von a ($syn_{ca} = 13$), während a durch die parallele Bearbeitung von c benachteiligt wird ($syn_{ac} = -14$).

4.1 Scheduling in der Trainingsphase

Aufgabe der Scheduling-Algorithmen ist die Auswahl einer Kombination von MPL-vielen Anfragen aus der Menge der wartenden Anfragen. Das Scheduling in der Trainingsphase verfolgt zum einen das Ziel, präzise Synergiewerte zu ermitteln. Zum anderen soll die Trainingsdauer gering sein, so dass schnell zur Optimierungsphase übergegangen werden kann. Um diese Ziele zu erreichen, können die folgenden Scheduling-Strategien angewendet werden:

Zufälliges Scheduling Beim zufälligen Scheduling werden zufällige Anfragekombinationen gewählt und in der Datenbank ausgeführt. Da die Ankunftsreihenfolge der Anfragen in der Datenbank zufällig ist, entspricht diese Strategie der Ausführung von Anfragen in der Ankunftsreihenfolge.

Minimale Unbekannte Ziel der Strategie ist die Füllung der Synergiematrix mit möglichst zuverlässigen Werten. Dazu werden bevorzugt die Kombinationen mit den wenigsten unbekannten Anteilen ausgeführt. Sind bei einem MPL von 3 z.B. bereits die Anteile s_{ab} und s_{bc} bekannt, so besitzt die Kombination abc lediglich einen unbekannten Anteil s_{ac} und wird daher einer Kombination mit mehreren Unbekannten vorgezogen.

Maximale Unbekannte Ziel der Strategie ist die möglichst schnelle Füllung der Synergiematrix. Im Gegensatz zur „Minimale Unbekannte“-Strategie können die erzeugten Werte jedoch relativ ungenau sein, weil sie nur auf wenigen verschiedenen Kombinationen basieren. Bevorzugt ausgeführt werden Kombinationen mit ausschließlich unbekannten Anteilen, danach Kombinationen mit nur einem bekannten Anteil etc.

Minimale Unbekannte, Minimale Kombinationen Die Präzision, mit der individuelle Anteile berechnet werden können, hängt stark von der Anzahl an unterschiedlichen Kombinationen, in denen der Anteil beobachtet wurde, ab. Die „Minimale Unbekannte, Minimale Kombinationen“-Strategie erweitert deshalb die „Minimale Unbekannte“-Strategie um die Betrachtung der Anfragekombinationen, in denen die Anteile auftreten. D.h. es werden Anfragekombinationen bevorzugt, welche noch nie oder möglichst selten ausgeführt wurden und die eine minimale Anzahl von unbekannten Anteilen enthalten. Dies stellt sicher, dass nicht wiederholt die gleichen Anfragekombinationen ausgeführt werden, weil diese keine neuen Gleichungen für das Gleichungssystem und damit keine zusätzlichen Informationen für die Synergieermittlung liefern.

Maximale Unbekannte, Minimale Kombinationen Analog zur „Minimale Unbekannte, Minimale Kombinationen“-Strategie führt die „Maximale Unbekannte, Minimale Kombinationen“-Strategie bevorzugt seltene Anfragekombinationen aus, allerdings solche mit einer möglichst großen Anzahl an unbekannten Anteilen.

4.2 Scheduling in der Optimierungsphase

In der Optimierungsphase wird der Datenbank-Workload auf Basis des in der Trainingsphase gewonnenen Wissens so effizient wie möglich ausgeführt. Das Scheduling in der Optimierungsphase verfolgt das Ziel, die ermittelten Synergien bestmöglich auszunutzen. Zusätzlich werden weiterhin Monitoring-Daten gesammelt, so dass unzureichende Informationen vervollständigt und langfristige Änderungen erkannt werden können. In der Optimierungsphase wird folgende Scheduling-Strategie verfolgt:

Maximale Synergie Die Synergie einer Kombination c setzt sich aus den Synergien der enthaltenen Anfragepaare zusammen und wird wie folgt berechnet:

$$syn_c = \sum_{s_{xy} \in c} \left(syn_{xy} + syn_{yx} \right)$$

Die Synergien der einzelnen Paare können sich dabei gegenseitig aufheben. Insgesamt besitzt jedoch diejenige Kombination den höchsten Synergiewert, die die meisten synergetischen Anfragen beinhaltet. Die Ermittlung eines optimalen Schedules für alle wartenden Anfragen ist auf Grund der exponentiellen Komplexität nicht effizient möglich. Die „Maximale Synergie“-Strategie verfolgt deshalb eine Greedy-Strategie und wählt die Anfragekombination mit der höchsten Synergie zur Ausführung in der Datenbank.

5 Evaluierung

Zur systematischen Evaluierung der vorgestellten Trainingsalgorithmen unter verschiedenen Rahmenbedingungen haben wir ein Simulationssystem entwickelt. Parameter wie MPL und Anzahl der Anfragetypen sowie deren Laufzeit und Wechselwirkungen untereinander können flexibel variiert werden. Das Simulationssystem basiert auf einer simulierten Datenbank. Anstelle einer realen Ausführung der Anfragen berechnet diese die Antwortzeit für eine Anfragekombination basierend auf einer Menge von vorgegebenen Synergiewerten. Die berechneten Zeiten werden vom Workload-Optimierer für den Aufbau bzw. die Vervollständigung der Synergiematrix verwendet. Die Qualität der Trainingsalgorithmen lässt sich damit leicht überprüfen, indem die ermittelten Synergiewerte mit den Vorgabewerten der simulierten Datenbank verglichen werden. Mit Hilfe der Simulation wird ermittelt, welcher der Algorithmen die Matrix während der Trainingsphase am schnellsten füllt und wie präzise die berechneten Wechselwirkungen sind. Desweiteren wird der Einfluss von Laufzeitschwankungen auf den Aufbau der Synergiematrix untersucht. Unsere erste Implementierung der Simulation dient der Nachvollziehung der vorgestellten Konzepte und wurde noch nicht im Hinblick auf die Skalierbarkeit untersucht.

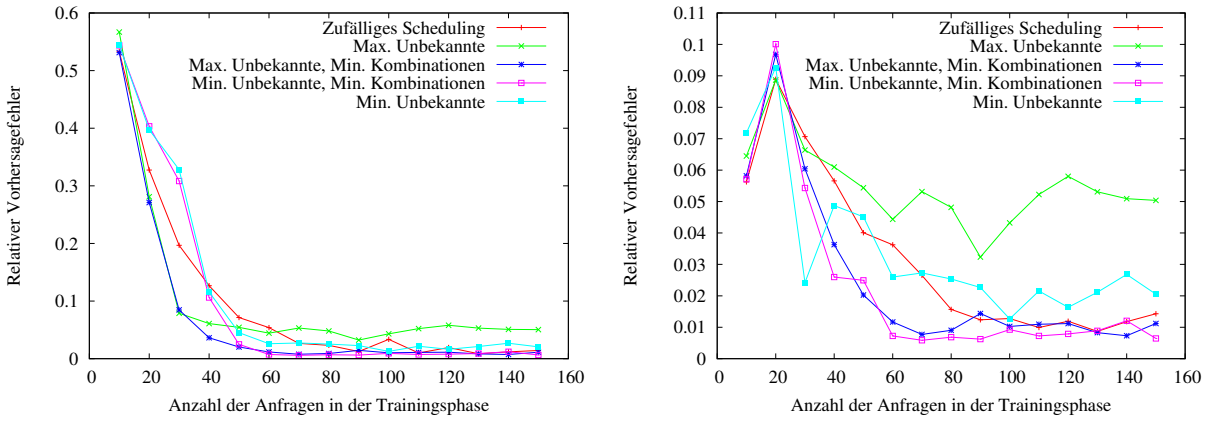
5.1 Trainingsalgorithmen

Als Qualitätsmaß für die Genauigkeit der Trainingsalgorithmen wird die Präzision der ermittelten Synergiewerte verwendet. Je geringer die Abweichung der für eine Kombination c vorhergesagten Ausführungszeit eRT_c von der tatsächlichen Ausführungszeit rRT_c ist, desto geringer ist der relative Fehlerkoeffizient von c (er_c):

$$er_c = \frac{|eRT_c - rRT_c|}{rRT_c}$$

Der relative Vorhersagefehler (*pError*) erfasst die durchschnittliche Genauigkeit der Vorhersage für alle möglichen Anfragekombinationen und ist definiert als:

$$pError = avg\left(er_c\right)$$



(a) Relativer Vorhersagefehler

(b) Relativer Vorhersagefehler, eingeschränkt auf bekannte Matrixfelder

Abbildung 4: Trainingsalgorithmen im Vergleich

Abbildung 4(a) zeigt für die verschiedenen Trainingsalgorithmen die Entwicklung des relativen Vorhersagefehlers *pError* bei steigender Dauer der Trainingsphase. Die Simulation wurde in diesem Experiment für einen MPL von 3 und 5 verschiedene Anfragetypen ähnlicher Komplexität durchgeführt, welche mit gleichverteilter Häufigkeit auftreten. Wie in der Abbildung zu sehen ist, erzielen alle Algorithmen eine gute Vorhersageleistung ab einer Anzahl von 40 Anfragen in der Trainingsphase.

Im Vergleich zum zufälligen Scheduling kann durch die gezielte Ausführung von Anfragekombinationen mit möglichst vielen Unbekannten („Maximale Unbekannte“ und „Maximale Unbekannte, Minimale Kombinationen“) die Gesamtgenauigkeit der Vorhersage gesteigert werden. Die vergleichsweise schlechte Leistung der Strategien „Minimale Unbekannte“ und „Minimale Unbekannte, Minimale Kombinationen“ lässt sich dadurch erklären, dass diese versuchen, zunächst möglichst wenige Synergiewerte möglichst genau vorherzusagen. Dadurch fällt der Gesamtfehler über die ganze Matrix gesehen relativ langsam ab. Dieser Effekt wird deutlich, wenn man nur die Felder der Matrix für die schon Messergebnisse vorliegen, betrachtet, wie in Abbildung 4(b) zu sehen.

In diesem Fall führen die beiden Strategien zu einer deutlich schnelleren Verbesserung der Vorhersage als bei zufälliger Ausführung. Die Schwankungen des relativen Vorhersagefehlers sind dadurch bedingt, dass leere Felder in der Matrix zunächst mit unpräzisen Werten gefüllt werden. Erst durch die Ausführung weiterer Kombinationen kann deren Genauigkeit erhöht werden.

Das ausgewogenste Verhalten was sowohl die Präzision der Vorhersage als auch die schnelle Füllung der Matrix angeht hat die Strategie „Maximale Unbekannte, Minimale Kombinationen“ weshalb sie in den folgenden Benchmarks zur Optimierungsphase als Trainingsalgorithmus verwendet wird.

Die Ausführungszeit von Anfragekombinationen in der Datenbank unterliegt in der Praxis Schwankungen, z.B. weil externe Prozesse Systemressourcen belegen oder der Datenbankpuffer besonders viele oder wenige der angefragten Seiten gespeichert hat. Um den Einfluss dieser Schwankungen auf die Trainingsalgorithmen quantifizieren zu können, wurde die simulierte Datenbank dahingehend erweitert, dass die berechnete Ausführungszeit für eine Anfragekombination in einem Bereich von ϵ Prozent um den wirklichen Wert schwanken kann. Abbildung 5 zeigt den relativen Vorhersagefehler in der Trainingsphase bei einem Wert von $\epsilon = 15\%$. Im Vergleich zur Ausführung ohne Schwankungen aus Abbildung 4(a) nimmt der Vorhersagefehler nur schwach zu. Durch wiederholte Ausführung der Anfragekombinationen und die Mittelwertbildung über die beobachteten Antwortzeiten können Schwankungen im Laufe der Zeit zuverlässig bei der Synergieberechnung kompensiert werden.

5.2 Optimierungsphase

Primäres Ziel der Optimierungsphase ist die Minimierung der Ausführungsdauer für alle wartenden Anfragen. Um vergleichbare Ergebnisse zu erhalten wurde ein Batch-Szenario zur Evaluation gewählt, in dem alle Anfragen zu Beginn der Optimierungsphase in der Warteschlange vorhanden sind. Es können also keine Abweichungen durch eine unterschiedliche Ankunftsreihenfolge der Anfragen auftreten, wie es bei interaktiven Workloads der Fall wäre.

Abbildung 6 zeigt die Gesamtdauer für die Ausführung verschieden großer Workloads. Zu sehen ist die Bearbeitungsdauer der „Maximale Synergie“-Optimierungsstrategie bei einer vorhergehenden Trainingsphase von 10, 30 und 50 Anfragen. Vergleicht man die Ausführung bei unterschiedlich langer Trainingsphase, wird ersichtlich dass durch die Feedback-Schleife in

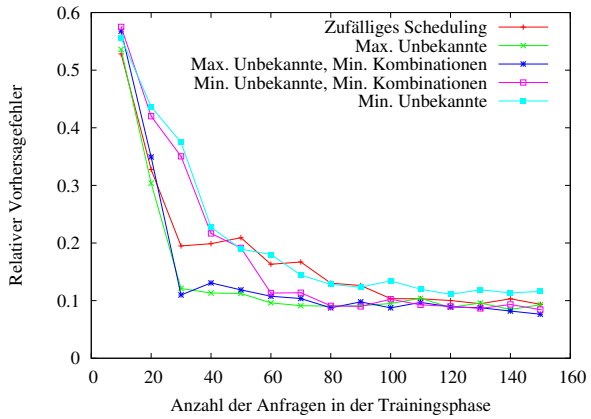


Abbildung 5: Schwankungen in der Ausführungszeit

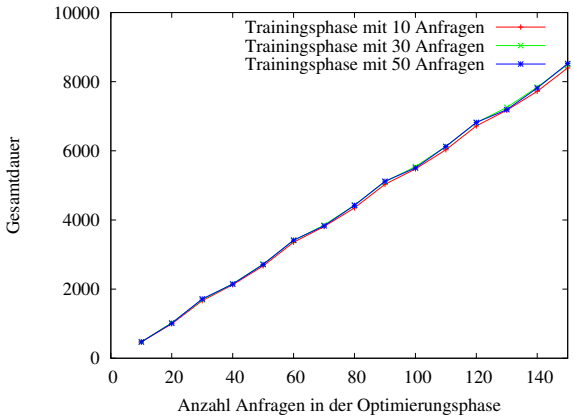


Abbildung 6: Scheduling in der Optimierungsphase

der Optimierungsphase fehlende Synergieinformationen im laufenden Betrieb ermittelt werden können. Die Gesamtausführungsdauer in der Optimierungsphase nimmt nur unmerklich zu wenn eine kurze Trainingsphase von 10 Anfragen statt einer Trainingsphase von 50 Anfragen gewählt wird.

6 Zusammenfassung und laufende Arbeiten

In dieser Arbeit wurde ein Monitoring-basierter Ansatz zur Optimierung von Datenbank-Workloads vorgestellt. Durch die Zerlegung der beobachteten Laufzeit von Anfrageblöcken in paarweise Einflusskomponenten und die Lösung des daraus entstehenden linearen Gleichungssystems können Synergien zwischen Anfragen abgeleitet werden. In einer anfänglichen Trainingsphase werden zunächst verschiedenartige Kombinationen ausgeführt, um ein möglichst breites Wissen über vorhandene Synergien zu erlangen und die Optimierung auf lokale Maxima zu vermeiden. In der anschließenden Optimierungsphase werden die erkannten Synergien ausgenutzt, um die Bearbeitung von Anfragen in der Datenbank effizient zu gestalten. Die dabei gewonnenen Erkenntnisse über Anfragelaufzeiten fließen über eine Feedback-Schleife wieder in die Ermittlung der Synergien ein. Mit Hilfe eines Simulationssystems wurden verschiedene Trainingsalgorithmen verglichen und die Effektivität der Feedback-Schleife in der Optimierungsphase belegt.

Nach der prinzipiellen Evaluation mit Hilfe des Simulationssystems, planen wir den Einsatz eines realen Datenbanksystems und die Evaluierung der möglichen Leistungsgewinne für verschiedene Workloads. Bei der Klassifikation der Anfragen wurde in dieser Arbeit davon ausgegangen, dass der Einfluss von Anfrageparametern auf die beobachtete Ausführungszeit vernachlässigbar ist. Sollten die Daten in der Datenbank nicht gleichverteilt sein, sondern eine starke Schieflage haben, trifft diese Annahme jedoch nicht zu [Reddy and Haritsa, 2005]. Eine Möglichkeit, die Vorhersagegenauigkeit in diesen Fällen zu erhöhen, besteht darin für verschiedene Parameterwerte eigene Anfragetypen einzuführen.

Literatur

[Ahmad et al., 2008] Ahmad, M., Aboulnaga, A., Babu, S., and Munagala, K. (2008). Modeling and Exploiting Query Interactions in Database Systems. In *CIKM '08: Proceedings of the ACM 17th International Conference on Information and Knowledge Management*, Napa, California, USA.

[Choenni et al., 1997] Choenni, S., Kersten, M. L., and van den Akker, J. F. P. (1997). A Framework for Multi Query Optimization. In *Proceedings of the 8^{th} International Conference on Management of Data (COMAD)*, pages 165–182, Madras, India.

[HP, 2007] HP (2007). HP NeoView Workload Management Services Guide.

[Krompass et al., 2007] Krompass, S., Kuno, H., Dayal, U., and Kemper, A. (2007). Dynamic Workload Management for Very Large Data Warehouses: Juggling Feathers and Bowling Balls. In *Proceedings of the 33^{rd} International Conference on Very Large Data Bases (VLDB)*, pages 1105–1115.

[Krompass et al., 2008] Krompass, S., Scholz, A., Albutiu, M.-C., Kuno, H. A., Wiener, J. L., Dayal, U., and Kemper, A. (2008). Quality of Service-enabled Management of Database Workloads. *IEEE Data Eng. Bull.*, 31(1):20–27.

[Markl et al., 2007] Markl, V., Haas, P. J., Kutsch, M., Megiddo, N., Srivastava, U., and Tran, T. M. (2007). Consistent Selectivity Estimation via Maximum Entropy. *The VLDB Journal*, 16(1):55–76.

[Microsoft, 2007] Microsoft (2007). Microsoft SQL Server 2005 Books Online. http://msdn2.microsoft.com/en-us/library/ms190419.aspx.

[Niu et al., 2006] Niu, B., Martin, P., Powley, W., Horman, R., and Bird, P. (2006). Workload Adaptation In Autonomic DBMSs. In *CASCON '06: Proc. of the 2006 Conf. of the Center for Advanced Studies on Collaborative Research*.

[O'Gorman et al., 2002] O'Gorman, K., Agrawal, D., and Abbadi, A. E. (2002). Multiple Query Optimization by Cache-Aware Middleware Using Query Teamwork. In *Proceedings of the 18^{th} International Conference on Data Engineering (ICDE)*, page 274.

[Oracle, 2001] Oracle (2001). The Oracle Database Resource Manager: Scheduling CPU Resources at the Application Level. http://research.microsoft.com/ jamesrh/hpts2001/submissions/.

[Reddy and Haritsa, 2005] Reddy, N. and Haritsa, J. R. (2005). Analyzing Plan Diagrams of Database Query Optimizers. In *Proceedings of the 31st international conference on Very large data bases (VLDB)*, pages 1228–1239.

[Schroeder et al., 2006] Schroeder, B., Harchol-Balter, M., Iyengar, A., Nahum, E., and Wierman, A. (2006). How to Determine a Good Multi-Programming Level for External Scheduling. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 60, Washington, DC, USA. IEEE Computer Society.

[Subramanian and Venkataraman, 1998] Subramanian, N. and Venkataraman, S. (1998). Cost Based Optimization of Decision Support Queries using Transient-Views. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 319–330.

[Zukowski et al., 2007] Zukowski, M., Héman, S., Nes, N., and Boncz, P. (2007). Cooperative Scans: Dynamic Bandwidth Sharing in a DBMS. In *Proceedings of the 33^{rd} International Conference on Very Large Data Bases (VLDB)*, pages 723–734.