Optimierte Speicherzuteilung an Operatoren in Datenbanksystemen

Jan Kristof Nidzwetzki

knidzwetzki@gmx.de

Abstract: Um Anfragen in DBMS effizient auszuführen, ist es entscheidend, vorhandenen Arbeitsspeicher zwischen Operatoren möglichst gut aufzuteilen. Erhält ein Operator zu viel Speicher, ist dieser vergeudet. Erhält ein Operator zu wenig Speicher, arbeitet er langsamer als möglich. Zudem profitiert nicht jeder Operator gleich stark von zusätzlichem Speicher. Diese Arbeit geht der Frage nach, wie die Charakteristik eines Operators, der Zusammenhang zwischen Laufzeit und benötigtem Arbeitsspeicher, durch eine Funktion approximiert werden kann.

1 Einleitung

Viele *Datenbankmanagementsysteme* (*DBMS*) lassen sich mit deskriptiven Abfragesprachen wie der *Structured Query Language* (*SQL*) ansprechen. Diese Sprachen beschreiben lediglich, wie ein Ergebnis auszusehen hat, nicht aber wie dieses berechnet wird [SSH11].

Es ist Aufgabe des *DBMS*, die Anfrage in einen *Ausführungsplan* zu übersetzen. Enthält ein *Ausführungsplan* mehrere *Operatoren*, so stehen diese in Konkurrenz um die Nutzung von Ressourcen. Das *DBMS* muss entscheiden, wie viel Speicher ein Operator nutzen darf [ND98] [HSS00]. Teilt man einem Operator mehr Arbeitsspeicher als benötigt zu, so ist er vergeudet. Teilt man einem Operator zu wenig Arbeitsspeicher zu, so arbeitet der Operator langsamer als möglich, da z. B. Daten auf die Festplatte ausgelagert werden müssen. Zudem profitiert nicht jeder Operator gleich stark von weiterem Arbeitsspeicher.

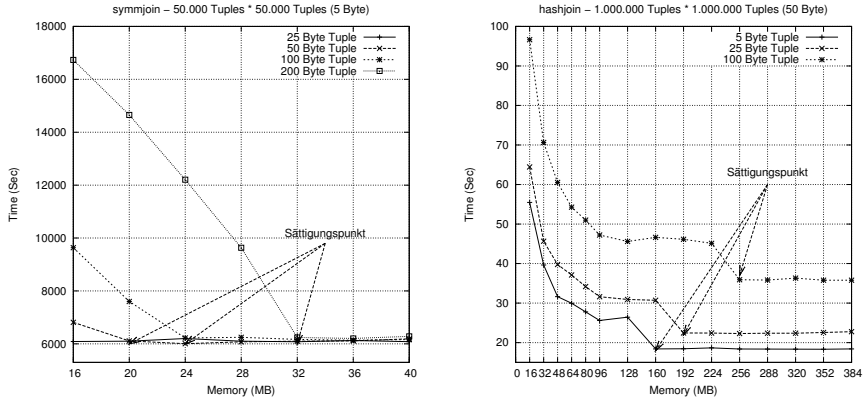
Grundsätzlich erwartet man zwei Reaktionen der Laufzeit bei einer höheren Arbeitsspeicherzuteilung: (1) der Operator kann den Arbeitsspeicher nutzen, um die Arbeit zu beschleunigen, (2) der zusätzliche Speicher kann nicht genutzt werden, da bereits alle Daten im Arbeitsspeicher vorliegen. Ziel des *DBMS* ist es, den Arbeitsspeicher so aufzuteilen, dass die Laufzeit des *Ausführungsplans* minimal wird. Ein tieferes Verständnis des Laufzeitverhaltens von Operatoren kann dem *DBMS* bei der Aufteilung des Speichers helfen.

An der *Fernuniversität in Hagen* ist ein frei verfügbarer DBMS-Prototyp mit dem Namen SECONDO entwickelt worden [SEC13] [GdAA+05]. Dieses *DBMS* wurde genutzt, um zu untersuchen, wie sich verschiedene Arbeitsspeicherzuteilungen auf die Laufzeit von Operatoren auswirken. Der in SECONDO enthaltene *Optimierer* kann für viele Operatoren zuverlässige Selektivitätsschätzungen abgeben. Diese Schätzungen ermöglichen es vor dem Ausführen einer Anfrage die Anzahl der zu verarbeitenden Tupel der Operatoren im *Ope-*

ratorbaum zu bestimmen. Zudem ist in SECONDO das Konzept der *Operator basierenden Fortschrittsschätzung* implementiert [Güt08]. Dieses Konzept ermöglicht es Operatoren, Schätzungen über ihre Laufzeit abzugeben. Grundlage hierfür bilden *Kostenmodelle*, die das Verhalten der Operatoren beschreiben.

2 Experimente

Für verschiedene Operatoren wurde im Rahmen dieser Arbeit untersucht, wie die Laufzeiten variieren, wenn die Größe des zugeteilten Arbeitsspeichers verändert wird [Nid12]. Die Experimente haben das erwartete Verhalten bestätigt. Bis zu einem gewissen Punkt kann der Operator von zusätzlichem Arbeitsspeicher profitieren und die Laufzeit sinkt. Nach diesem Punkt bleibt die Laufzeit konstant und der Arbeitsspeicher ungenutzt. Der Punkt, an dem dieser Übergang stattfindet, wird im Folgenden als *Sättigungspunkt* bezeichnet.



(a) Laufzeitverhalten des Join-Operators symmjoin (Variante eines Nested-Loop-Join)

(b) Laufzeitverhalten des Join-Operators hashjoin

Abbildung 1: Laufzeitverhalten zweier Join-Operatoren mit verschiedenen Eingabegrößen.

Die Experimente haben gezeigt, dass die Operatoren auf zwei Arten auf weiteren Arbeitsspeicher reagieren. Einige weisen ein lineares Verhalten zwischen Arbeitsspeicher und Laufzeit auf (Abb. 1(a)), andere ein antiproportionales (Abb. 1(b)). Liest man die Abbildung 1(b) von rechts nach links, so sieht man, dass links des *Sättigungspunktes* die Laufzeit sprunghaft ansteigt und zunächst nur eine sehr leichte Steigung aufweist. Der sprunghafte Anstieg lässt sich mit der Implementation der Operatoren erklären. Beim ersten Tupel, welches nicht mehr im Arbeitsspeicher gehalten werden kann, müssen ggf. Daten auf die Festplatte ausgelagert werden und der Operator verändert seinen Algorithmus.

Die Auswertung der Experimente hat ergeben, dass sich das Laufzeitverhalten sehr gut durch die beiden Funktionen $mx + b$ und $\frac{h}{x} + b$ approximieren lässt (Abb. 2). Die Variablen m, h und b stellen Lageparameter dar, die Variable x steht für den zugeteilten Arbeitsspeicher.

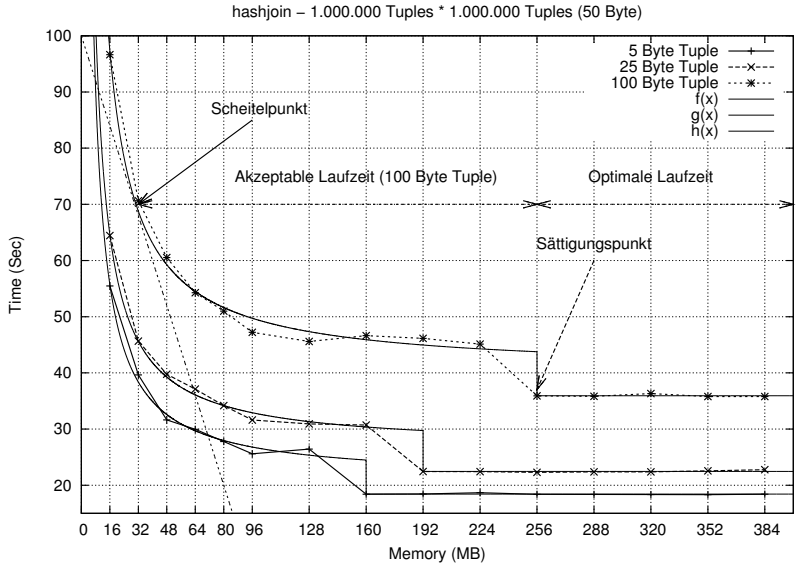


Abbildung 2: Bereich der akzeptablen Laufzeit und der optimalen Laufzeit für einen Join-Operator. Die Funktion $\frac{h}{x} + b$ wurde zur Approximation genutzt. Die parametrisierten Funktionen f, g und h approximieren das Verhalten des Operators bei bestimmten Eingabegrößen. Die optimale Laufzeit liegt rechts des Sättigungspunktes vor. Die akzeptable Laufzeit links des Sättigungspunktes.

Für jede Anwendung müssen diese Funktionen korrekt parametrisiert werden. Durch den *Optimierer* von SECONDO kann abgeschätzt werden, wie viele Tupel jeder Operator zu verarbeiten hat. Die *Fortschrittsschätzung* erlaubt es abzuschätzen, wie lange die Berechnung dauert. Die Schätzungen ermöglichen es, die Lageparameter zu berechnen und dem *DBMS* die parametrisierte Funktion für weitere Optimierungen zur Verfügung zu stellen.

Bei vielen Operatoren kann Arbeitsspeicher eingespart werden, ohne dass die Laufzeit sonderlich stark ansteigt. In Abbildung 2 steigt die Laufzeit von 38 Sekunden im *Sättigungspunkt*, um ca. den Faktor zwei, auf 73 Sekunden im *Scheitelpunkt* an. Der Arbeitsspeicher kann jedoch um den Faktor acht, von 256 MB im *Sättigungspunkt* auf 32 MB im *Scheitelpunkt*, reduziert werden.

3 Fazit

Die beiden vorgestellten Funktionen ermöglichen es, das Laufzeitverhalten von Operatoren gut zu approximieren. Die Kenntnis einer solchen Funktion hilft, den Arbeitsspeicher effizient zuzuteilen. Hierdurch wird es dem *DBMS* möglich, genau den Operatoren mehr Arbeitsspeicher zuzuweisen, die am stärksten davon profitieren. Da andere *DBMS* ähnlich wie SECONDO arbeiten, ist zu erwarten, dass in diesen ebenfalls eine Approximation der Laufzeit möglich ist. Dies sollte in einer weiteren Arbeit genauer untersucht werden.

Literatur

[GdAA+05] Ralf Hartmut Güting, Victor Teixeira de Almeida, Dirk Ansorge, Thomas Behr, Zhiming Ding, Thomas Höse, Frank Hoffmann, Markus Spiekermann und Ulrich Telle. SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, Seiten 1115–1116. IEEE Computer Society, 2005.

[Güt08] Ralf Hartmut Güting. Operator-Based Query Progress Estimation. In *Informatik Berichte*, number 343. Fernuniversität in Hagen, Feb 2008.

[HSS00] A Hulgeri, S Sudarshan und S Seshadri. Memory cognizant query optimization. Proceedings of COMAD, 2000.

[ND98] Biswadeep Nag und David J. DeWitt. Memory allocation strategies for complex decision support queries. In *Proceedings of the seventh international conference on Information and knowledge management*, CIKM '98, Seiten 116–123, New York, NY, USA, 1998. ACM.

[Nid12] Jan Kristof Nidzwetzki. *Operator-Kostenmodelle für Fortschrittsschätzung und Optimierung in Datenbanksystemen*. Bachelorarbeit, Lehrgebiet Datenbanksysteme für neue Anwendungen, Fernuniversität in Hagen, 2012.

[SEC13] SECONDO Webseite, 2013. http://dna.fernuni-hagen.de/Secondo.html/ - Abgerufen am 10.01.2013.

[SSH11] Gunter Saake, Kai-Uwe Sattler und Andreas Heuer. *Datenbanken - Implementierungstechniken, 3. Auflage*. mitp-Verlag, Redline GmbH, 2011.