I/O-Parallelität in Datenbanksystemen Entwurf, Implementierung und Evaluation eines Speichersystems für Disk-Arrays

Author(s):
Zabback, Peter

Publication Date:
1994

Permanent Link:
https://doi.org/10.3929/ethz-a-000943544 →

Diss. ETH Nr. 10629

I/O-Parallelität in Datenbanksystemen

Entwurf, Implementierung und Evaluation eines Speichersystems für Disk-Arrays

ABHANDLUNG

zur Erlangung des Titels eines
Doktors der Technischen Wissenschaften
der
EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE
ZÜRICH

vorgelegt von
PETER ZABBACK
Dipl.-Inform. TH Darmstadt
aus Frankfurt a.M.

Angenommen auf Antrag von
Prof. Dr. G. Weikum, Referent
Prof. Dr. P. Scheuermann, Korreferent

Kurzfassung

Die vorliegende Arbeit untersucht das Problem der Parallelisierung von I/O-Aufträgen in Datenbanksystemen. Durch die in den letzten Jahren rasant zunehmende Prozessorgeschwindigkeit wird das I/O-System mehr und mehr zur leistungslimitierenden Komponente eines Rechners. Ein vielversprechender Ansatz, dieser Entwicklung zu begegnen, ist der Einsatz von Parallelität.

Um I/O-Parallelität überhaupt nutzen zu können, sind die Anwendungsdaten zu partitionieren und die entstehenden Partitionen über die Platten des Systems zu verteilen. In einem ersten Schwerpunkt dieser Arbeit untersuchen wir daher die Zielkonflikte bei der Wahl eines möglich günstigen Partitionierungsgranulats. Zwar kann durch den Einsatz von Parallelität die Antwortzeit einzelner I/O-Aufträge reduziert werden, diese Verbesserung wird aber durch einen erhöhten Ressourcenverbrauch erkauft, da nun mehrere Platten an der Bedienung eines einzelnen Auftrags beteiligt sind. Das Verhalten einer einzelnen Platte ist mittels eines mathematischen Warteschlangenmodells gut faßbar und wohlverstanden. Für den hier betrachteten allgemeineren Fall, bei dem mehrere Platten an der Bedienung eines I/O-Auftrags beteiligt sind, ist keine mathematisch geschlossene Lösung bekannt. Wir haben daher eine approximative Lösung dieses Problems hergeleitet, die auf Mittelwertbetrachtungen der Auftragsgrößen und Abschätzungen über die mittlere Ankunftsrate von I/O-Aufträgen basiert. Es zeigt sich, daß diese Approximation die tatsächlichen Verhältnisse mit ausreichender Genauigkeit beschreibt. Von diesem Modell haben wir Algorithmen abgeleitet, die bei vorgegebener I/O-Systemkonfiguration und bekannten Lastcharakteristika für jedes Segment (Datei) das jeweils günstigste Partitionierungsgranulat bestimmt. Das Modell kann auch verwendet werden, um bei gegebenen Lastcharakteristika und Leistungsanforderungen das I/O-System zu konfigurieren.

Für die Partitionierungsalgorithmen sind wir von der Annahme ausgegangen, daß die I/O-Last möglichst gleichmäßig über die Platten des Systems verteilt ist. Wegen der im allgemeinen ungleichmäßigen Verteilung der Zugriffshäufigkeiten auf die einzelnen Partitionen ist eine gute Lastbalancierung jedoch nicht ohne zusätzliche Maßnahmen zu erreichen. Die Arbeit beschreibt daher in einem zweiten Schwerpunkt Allokationsverfahren zur Lastbalancierung, die wegen der Komplexität des Problems — das allgemeine Allokationsproblem ist NP-hart — auf Heuristiken basieren.

Das Lastaufkommen in einem Rechnersystem ist nicht statisch und unterliegt ständigen Fluktuationen. Wir haben daher einen Algorithmus entwickelt, der die Häufigkeitsverteilung der I/O-Aufträge auf die Partitionen beobachtet und im laufenden Betrieb durch die Verlagerung einzelner Partitionen versucht, die I/O-Last möglichst gleichmäßig über alle Platten zu verteilen.

Sowohl die Algorithmen zur Datenpartitionierung als auch zur Lastbalancierung haben

wir prototypisch implementiert und einer systematischen Evaluation unterzogen. Diese Evaluation erfolgte mit Hilfe synthetischer und realer I/O-Lasten aus verschiedenen Anwendungsbereichen. Dabei hat sich die segmentspezifische Partitionierung gegenüber globalen Partitionierungsverfahren, bei denen alle Segmente mit dem gleichen Granulat partitioniert werden, als vorteilhaft erwiesen. Durch die individuelle Anpassung der Segmentpartitionierung an die jeweiligen Anforderungen der I/O-Aufträge ist eine deutliche Reduzierung der mittleren Antwortzeit erzielbar.

Die Evaluation zeigt auch, daß sich durch die Lastbalancierungsverfahren signifikante Laufzeitgewinne erzielen lassen. Insbesondere der dynamische Lastbalancierungsalgorithmus hat sich als sehr robust erwiesen und ist in der Lage, Engpaßsituationen frühzeitig zu erkennen und durch geeignete Maßnahmen zu beseitigen.

Abstract

This thesis investigates the problem of I/O parallelism in database management systems. Due to the dramatically increasing processor speeds, more and more applications will become I/O limited. I/O parallelism is being considered as a promising approach to overcome this trend.

In order to exploit I/O parallelism the application's data has to be partitioned and the partitions have to be allocated across the disks of the I/O system. Thus, we discuss the main tradeoffs in choosing an appropriate granule (i.e., striping unit) for the data partitioning. I/O parallelism reduces the response time of individual I/O requests. But this improvement is achieved at the expense of increased resource consumption, since each I/O request is served by multiple disks. The scenario where each request is served by a single disk is well understood and can be modeled via a queueing model. However, no analytical model is known for the general case of I/O requests which are served by multiple disks. Thus, our approach is an approximation, which takes into account the average request size of each segment and the throughput requirements of the application. The accuracy of our model is sufficient to describe the effects of I/O parallelism on the response time and throughput of the I/O requests. We used this model to develop algorithms for choosing the striping unit of files, based on the assumption that workload characteristics are given.

We assumed for the partitioning problem that the I/O load is distributed uniformly across the disks of the I/O system. Almost all applications exhibit a significant skew in their access pattern, i.e., some files are accessed much more frequently than other files. Thus, the load balance of the I/O system depends on the placement of the data. This data placement problem is known to be NP-hard. Hence, our solution is based on a heuristic approach. The allocation algorithm distributes the I/O-load as even as possible across all disks by a greedy heuristic.

Many applications exhibit dynamic load changes that may render an originally good allocation into an unbalanced system. In order to deal with evolving file access patterns, we developed an algorithm that can redistribute the load by migrating data from one disk to another disk whenever a significant imbalance of the load is detected.

We have built a file system prototype that allows for the striping of files on an individual basis and incorporates the heuristic algorithms for dynamic load balancing. We performed a comprehensive performance evaluation of our methods based on synthetic as well as real trace data from various applications. The performance study shows the advantage of file specific partitioning, compared to global methods where each file is partitioned by the same striping unit. In addition the performance study shows the benefits of the dynamic load balancing algorithm.