- Technologie für Parallele Datenbanken -

Bericht zum Workshop

Bernhard Mitschang
SFB 342, Teilprojekt B2 “Parallelisierung von Datenbanksystemen”

Zusammenfassung

Der vorliegende Bericht dient als Dokumentation zum Workshop “Parallele Datenbank-Technologie”, abgehalten am 8. und 9. August 1996 an der TU München. Das übergeordnete Ziel des Workshops war es, aktuelle und neue Themen vorrangig aus dem Bereich Paralleler Datenbanksysteme zu diskutieren. Dieser Bericht umfaßt ein Inhaltsverzeichnis mit Themenübersicht, eine Zusammenstellung der Kurzfassungen zu den einzelnen Vorträgen, eine kurze Abschlußdiskussion sowie auch eine vollständige Teilnehmerliste. Über die angegebenen Adreßinformationen kann weitere Literatur zu den wissenschaftlichen Arbeiten der einzelnen Teilnehmer bezogen werden.

Inhaltsverzeichnis

1 Vorwort

Das Teilprojekt B2 "Parallelisierung von Datenbanksystemen" im Sonderforschungsbereich 342 mit dem Titel "Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen" befaßt sich mit Konzept- und Implementierungsfragen zu Parallelen Datenbanksystemen und hat mittlerweile einen lauffähigen Prototyp, das Parallele Datenbanksysteme MIDAS [BJLM96], entwickelt. MIDAS dient als sog. Testbett für detaillierte Untersuchungen diverser Parallelisierungstechniken und Implementierungsfragen in Parallelen Datenbanksystemen. Mit der Fertigstellung einer ersten MIDAS-Version war es uns sehr wichtig, die dort realisierten Konzepte mit anderen Fachleuten auf diesem Gebiet zu diskutieren. Aus diesen Grunde haben wir verschiedene Arbeitsgruppen zu einem gemeinsamen Treffen und Workshop nach München eingeladen.

Das übergeordnete Ziel des Workshops war es, aktuelle und neue Themen vorrangig aus dem Bereich Paralleler Datenbanksysteme zu diskutieren, einen Überblick über die Arbeiten der anderen Gruppen zu bekommen, Kontakte dorthin aufzubauen bzw. bestehende aufzufrischen und auch die eigenen Arbeiten und Ideen vorzustellen. Um den Diskussionscharakter des Workshops zu betonen, haben wir speziell auch darum gebeten, neue Forschungsthemen und wichtige Arbeitsgebiete vorzustellen und zu diskutieren.

Der Workshop wurde zu Semesterende am 8. und 9. August an der TU München abgehalten. Insgesamt haben sich trotz kurzfristiger Ankündigung und Terminfestsetzung 30 Teilnehmer eingefunden, worunter sich erfreulicherweise auch 3 Industrievertreter befanden.

Der vorliegende Bericht dient als Dokumentation zum abgehaltenen Workshop und umfaßt ein Inhaltsverzeichnis mit Themenübersicht, eine Zusammenstellung der abgegebenen Kurzfassungen zu den einzelnen Vorträgen sowie auch eine vollständige Teilnehmerliste. Über die angegebenen Adressinformationen kann weitere Literatur zu den wissenschaftlichen Arbeiten der einzelnen Teilnehmer bezogen werden.

Die sofortige Bereitschaft der angesprochenen Arbeitsgruppen, sich an diesem Workshop zu beteiligen, sowie Tiefe und auch Breite der behandelten Diskussionsthemen lassen ein großes Interesse an der Thematik Paralleler Datenbanksysteme erkennen. Der Erfolg des Workshops zeigt sich nicht zuletzt auch dadurch, daß schon verabredet wurde, einen Folge-Workshop durchzuführen.

Für das zahlreiche Erscheinen und die Diskussionsbreitschaft bedanke ich mich bei allen Teilnehmern recht herzlich. Den lokalen Organisatoren gebührt mein Dankeschön für die schnelle Vorbereitung und reibungslose Durchführung des Workshops. Insgesamt hat es sich gezeigt, daß es auch mit geringem Aufwand möglich ist, interessante Forschungsthemen erfolgreich und auf hohem Niveau mit einem Fachpublikum zu behandeln.

München, im November 1996

B. Mitschang

Literatur:

[BJLM96] Bozas, G., Jaedicke, M., Listl, A., Mitschang, B., et al.: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project, Proc. EUROPAR, Lyon, France, in: LNCS 1124, Springer Verlag, pp. 881-886, 1996.

2 Architekturen und Ablaufkonzepte

2.1 Implementierung eines Parallelen DBMS auf einem Workstationnetz unter Verwendung einen existierenden DBMS (M. Exbrayat, Lyon)

Last years have seen the arising of parallel techniques into Database Management Systems (DBMS), in order to provide quick access to large and very large databases to many simultaneous users. Most of research has been driven through the assumption that parallel DBMS would run on Massively Parallel Machines (MPM). It must be noticed that such machines, even bringing an incomparable rise of performance, are still quite few, and represent a big investment. This brought hybrid architectures, such as workstation clusters, or networks of workstations to come to front page of research. More than suggestive aspects, such as global cost, it can be considered that workstations are widely used by many companies. It provides a satisfying robust and extensible computing power comparing to most parallel machines on the market.

This allows us to believe that virtual parallelism between small- to middle-size computers is a promising domain of investigation. Nevertheless, we must notice that any implementation drives to the use of a new DBMS. Our direction is then to allow users of classical Relational DBMS to extend the abililities of their existing system rather than changing for a brand-new one. We use a software extension that detects the level of use of the dedicated machine on which the DBMS is running. The extension decides, according to the machine load, if a part of the work should be accomplished by other machines available over the Local Area Network. Only complex queries are parallelised, as selections and projections are left to the DBMS.

Our system uses two kinds of machines. The first one is the DBMS machine, including the test and parallelising extension. The second one consists of the other machines of the network, which we call "calculators". The extended DBMS contains a query interpreter, an optimizer, a calculator manager and a data distribution manager. The calculator manager evaluates the load of the machines deciding if they can be used as calculators. The data distribution manager gives informations concerning the data dispatched on the network, which are supposed to move depending on the machines' availability. As we offer a temporary parallel extension, we don't propose a permanent data distribution. Data is extracted from the DBMS files whenever it is needed. We choose to use a "SELECT" operation using cursors, rather than a direct access to data files, in order to allow a satisfying portability. Data stored in cursors are transmitted to the calculators as buckets of fixed size. Calculators are composed of a communication interface, a calcul unit, and a storage unit. Data and instructions are transmitted separately. This allows the calculators both to keep data after any execution, and to use a kind of pipelining over data buckets.

The research points introduced by this prototype are the mobility of data over the workstations, and the corresponding optimal parallel execution, as well as the load threshold concerning both the DBMS and the calculators.

The most important cost should be introduced by transactions, as they force the distributed data to be updated. As our system is bound to be used for document applications, where transactions are quite rare, we believe this system to allow satisfying performances.

2.2 Überblick über das PDBMS MIDAS (G. Bozas, S. Zimmermann, TU München)

Dieser Vortrag gibt eine Übersicht über den Systemaufbau des parallelen, relationalen, Shared-Disk Datenbanksystems MIDAS. Die einzelnen Systemkomponenten werden kurz vorgestellt und danach wird auf aktuelle Implementierungsvarianten und Probleme in den Bereichen Cache- und Sperrverwaltung und die parallele Query-Verarbeitung eingegangen. Bei der Cache- und Sperrverwaltung konzentrieren sich die Untersuchungen auf die Bewertung und Auswahl verschiedener Algorithmen bzgl. Leistung, Skalierbarkeit und Kombinierbarkeit. Bei der Query-Verarbeitung ist ein geeignetes Modell für die Kontrolle der parallelen Abläufe in MIDAS gesucht. Die Anforderungen an eine solche Komponente werden vorgestellt.

2.3 Betriebssystemeinbettung des parallelen Komplex-Objekt-DBMS PRIMA (M. Gesmann, Univ. Kaiserslautern)

Parallele Anfrageverarbeitung in Komplex-Objekt und objektorientierten DBMS wurde bisher kaum untersucht, da die dafür notwendigen Voraussetzungen in Form deskriptiver und mengenorientierter Anfragesprachen nicht gegeben waren. Erst mit der Standardisierung entsprechender Sprachen, wie sie zur Zeit zum Beispiel für OQL durch die ODMG oder für SQL3 erfolgen, gewinnt die parallele Verarbeitung in solchen DBMS an Bedeutung.

In diesem Beitrag werden zunächst die wesentlichen Unterschiede zwischen der relationalen und der Komplex-Objekt-Anfrageverarbeitung erläutert. Daran anschließend wird eine Systemarchitektur für das PRIMA DBMS vorgestellt, die auf einer Client/Server-Zerlegung basierend eine im Vergleich zu relationalen Systemen verhältnismäßig feingranulare Parallelität ausnutzt. Es werden verschiedene Abbildungen dieser Architektur auf Prozesse eines darunterliegenden Betriebssystems vorgestellt und diskutiert.

Auf dem OO7-Benchmark basierende Messungen zeigen, daß es keine optimale Abbildung für alle möglichen Lasten gibt. Allerdings zeigt eine der vorgestellten Abbildungen in nahezu allen Situationen ein sehr gutes Leistungsverhalten, so daß diese Abbildung in späteren Untersuchen mit dem PRIMA-System für alle Lasten genutzt werden kann. Neben dem Einfluß die die Wahl der Abbildung auf das Leistungsverhalten zeigt, wird durch die Messungen auch deutlich gezeigt, daß die Wahl der Verarbeitungsstrategien und der Systemschnittstellen einen deutlich höheren Einfluss auf die Antwortzeiten hat.

Abschließend sollen Überlegungen zu einer alternativen Implementierung des Systems auf der Basis neuerer Betriebssystemkonzepte, z. B. Threadkonzepte, verteilter gemeinsamer Speicher, direkte Abbildung von Dateien in den Adressraum eines Prozesses oder feste Allokation von Speicherbereichen im Hauptspeicher eines Rechners, vorgestellt werden.

Literatur:

Gesmann, Michael: Performance Evaluation of the Remote Cooperation System in PRIMA, Proc. 3rd Int. Conf. on Parallel and Distributed Systems, Austin, 1994, pp. 257-260

Gesmann, Michael: Mapping a Parallel Complex-Object DBMS to Operating System Processes, EUROPAR '96, Workshop: Parallel and Distributed Database System, Lyon, 1996

2.4 Synchronization in KBMS - Requirements and Solutions (T. Härder, Univ. Kaiserslautern)

As ever-larger knowledge bases (KBs) are being built, knowledge sharing becomes an aspect of paramount importance in Knowledge Base Management Systems (KBMSs). At first, we identify the concepts and operations such as user-defined and system-controlled relationships, rules, as well as methods to be supported by a concurrency control algorithm in KBMS. In the following, we outline important approaches used in relational, object-oriented, and complex-object database systems. A performance-determining factor of locking approaches is the efficient support of multiple locking granules, that is, their ability to apply implicit locking to a set of objects. To exploit implicit locking as far as possible, we propose the LARS (Locks using Abstraction Relationships' Semantics) approach for concurrency control in KBs. LARS synchronizes transactions through many different granules of locking, which are based on the semantics of the abstraction relationships commonly used in knowledge representation approaches. LARS supports a higher degree of potential concurrency in that it maintains different logical partitions of a KB graph, a means for representing KBs, and offers many lock types to be used on the basis of each one of the partitions. By such a way, LARS captures more of the semantics contained in a KB, through an interpretation of the (abstraction) relationships between objects, profits from such semantics for synchronizing the transactions, and thus makes feasible the exploitation of the inherent parallelism in a knowledge representation approach. Furthermore, we discuss the way of locking dynamically defined complex objects in the framework of the LARS approach.

2.5 Dynamic Load Balancing in Parallel Database Systems. (E. Rahm, Univ. Leipzig)

Dynamic load balancing is a prerequisite for effectively utilizing large parallel database systems. Load balancing at different levels is required in particular for assigning transactions and queries as well as subqueries to nodes. Special problems are posed by the need to support both inter-transaction/query as well as intra-transaction/query parallelism due to conflicting performance requirements. We compare the major architectures for parallel database systems, Shared Nothing and Shared Disk, with respect to their load balancing potential. For this purpose, we focus on parallel scan and join processing in multi-user mode. It turns out that both the degree of query parallelism as well as the processor allocation should be determined in a coordinated way and based on the current utilization of critical resource types, in particular CPU and memory.

3 Datenverteilung und Lastbalancierung

3.1 Datenverteilung als Grundlage von Skalierbarkeit in Workflow-Management-Systemen (H. Schuster, Univ. Erlangen-Nürnberg)

Das Ziel von Workflow-Management ist die Unterstützung der Ausführung von Geschäftsprozessen in Unternehmen beliebiger Größe und Struktur. Workflow-Management-Systeme (WFMS) implementieren eine geeignete Ausführungsumgebung für Workflows, d.h. computerunterstützt ausführbare Geschäftsprozesse. In Abhängigkeit vom konkreten Unternehmen, in dem ein WFMS eingesetzt wird, muß ein WFMS auf die gegebenen Verteilungsaspekte und auf veränderliche Lastsituationen reagieren, um eine adäquate Leistung zu erbringen. Ein monolithisches zentralisiertes WFMS (Ähnlich einem zentralen Datenbankserver) ist hierzu nicht in der Lage. Es wird weder den Verteilungsanforderungen gerecht, verschiedene örtlich getrennte Unternehmensbereiche können z.B. auf einem 'eigenen' WFMS bestehen, noch kann es ohne größere Anstrengungen an steigende Last angepaßt werden. Somit hat Skalierbarkeit bzgl. WFMS zwei Dimensionen: zum einen die Integration von neuen Unternehmensbereichen und zum anderen variierende Lastanforderungen. Deshalb ist ein verteiltes WFMS zu bevorzugen, das aus mehreren miteinander kooperierenden Servern besteht.

Die Grundlage für ein Kooperationsschema zwischen WFMS-Servern ist durch organisatorische Regeln gegeben. Sicherheits-, Strukturierungs- und Optimierungsgesichtspunkte in einem Unternehmen bestimmen, für wen ein gegebener WFMS-Server Dienste erbringen kann bzw. soll. Diese Regeln werden in der organisatorischen Relation zusammengefaßt. Diese Relation bildet die Grundlage für eine Datenallokations- und Replikationsstrategie. Entsprechend dem Need-to-know Prinzip werden Schemadaten für alle Serverinstanzen repliziert, was lokalen Zugriff auf diese Daten ermöglicht. Der Synchronisationsaufwand beim Zugriff auf Schemadaten ist gering, da Änderungen selten sind. Die zur Laufzeit anfallenden Beschreibungsdaten von Workflows (Instanzdaten) sind dagegen höchst änderungsanfällig. Um den Zugriff auf diese Daten zu optimieren, werden Instanzdaten partitioniert und auf verschiedene Workflow-Server verteilt, so daß die Daten einer Workflowinstanz von genau einem Workflow-Server bearbeitet werden. Somit werden schreibende Zugriffe von verschiedenen Servern auf dasselbe Instanz-datum vermieden. Zusätzlicher Synchronisationsaufwand wegen der Verteilung von WFMS-Servern kann also vermieden werden. Als Konsequenz dieser Vorgehensweise können prinzipiell beliebig viele WFMS-Server ohne nennenswerten Synchronisationsaufwand zu einem verteilten skalierbaren WFMS zusammengeschlossen werden.

Der kritische Punkt einer Implementierung für das obige Vorgehen ist das Auffinden des verantwortlichen Servers für eine gegebene Workflowinstanz. Ist dieser Suchvorgang zu teuer bzw. wächst der Aufwand stark mit der Anzahl der Workflow-Server, so stellt er einen ernstzunehmenden Engpaß dar. Aus diesem Grund wird ein direkter Adressierungsmechanismus vorgestellt, der eine verteilte, schwach konsistente Datenbank als Grundlage verwendet.

3.2 Konfiguration verteilter Workflow-Systeme unter Leistungs- und Verfügbarkeitsanforderungen (J. Weissenfels, Univ. Saarbrücken)

Workflow-Management-Systeme (WFMS) erobern immer weitreichendere Anwendungsgebiete, z.B. werden sie in Banken, Versicherungen und anderen Unternehmen eingesetzt. Zu den typische Anforderungen, die an unternehmensweite WFMS gestellt werden, gehören insbesondere Skalierbarkeit und hohe Verfügbarkeit. Um diese Anforderungen zu erfüllen ist eine sorgfältige Konfiguration des WFMS unbedingt notwendig.

In diesem Beitrag werden Erfahrungen bezüglich der Konfiguration eines verteilten unternehmensweiten WFMS aus dem Mentor-Projekt [WWW96] vorgestellt. In dem Projekt wurde als Middleware Komponente der TP-Monitor Tuxedo gewählt, der für die zuverlässige Nachrichtenübermittlung und Transaktionsverwaltung bei der verteilten Ausführung von Workflows verwendet wird. Zusätzlich übernimmt er die Prozessverwaltung des WFMS und ist damit erste Adresse für Konfigurationsüberlegungen in Mentor.

Zunächst wird die verteilte Ausführung eines Workflows durch kooperierende WF-Engines skizziert. Anschliessend wird die Struktur des TP-Monitors und die Integration der WFMS-Komponenten als Tuxedo-Prozesse vorgestellt. Im folgenden werden die Möglichkeiten des TP-Monitors zur Lastverteilung und Erhöhung der Verfügbarkeit des WFMS betrachtet. Aus diesen Überlegungen ergibt sich eine erste Ad-hoc Konfiguration für das System, die abschliessend unter Leistungs- und Verfügbarkeitsgesichtspunkten analysiert und erweitert wird.

Literatur:

[**WWW96**] Wodtke, D., Weissenfels, J., Weikum, G., Kotz-Dittrich, A.: The Mentor Project: Steps Towards Enterprise-Wide Workflow Management In: Proc. of the 12th IEEE International Conference on Data Engineering, New Orleans, LA, March 1996

3.3 Adaptive Datenreplikation durch Synchronisation mit "virtuellen Primärkopien" (R. Lenz, Univ. Erlangen-Nürnberg)

Das Ziel der Datenreplikation in geographisch verteilten Datenverwaltungssystemen ist die Erhöhung der Verfügbarkeit und die Verbesserung der Performance. Eine Verbesserung der Performance wird dabei in erster Linie durch eine erhöhte Zugriffslokalität erreicht. Die Sicherstellung des traditionellen Korrektheitskriteriums One-Copy Serialisierbarkeit, zur Synchronisation transaktionaler Zugriffe auf replizierte Daten, wirkt der Erreichung der gesetzten Ziele jedoch entgegen. Grund dafür ist die Forderung der Verteilungstransparenz sowie die uneingeschränkte Übertragung der ACID-Eigenschaften von Transaktionen in zentralisierten Systemen auf globale Transaktionen in verteilten Systemen.

In den vergangenen Jahren wurden viele Vorschläge gemacht, von diesen strengen Konsistenzvorschriften abzuweichen und durch Abschwächung der Konsistenzanforderungen eine erhöhte Verfügbarkeit und verbesserte Performance zu erreichen. Erste Ansätze in diese Richtung waren das Snapshot-Konzept und die sogenannten Quasi-Kopien. Bei beiden Methoden wird ein Replikat zu einer Primärkopie in eine Konsistenzbeziehung gesetzt. Der Unterschied besteht darin, daß beim Snapshot eine ereignisorientierte Aktualisierung spezifiziert wird, während bei der Quasi-Kopie eine kontinuierliche Aktualisierung durch Spezifikation eines maximal tolerierbaren Abstands zugrundegelegt wird. Beide Ansätze erlauben zwar eine Abschwächung von Konsistenzanforderungen, ermöglichen jedoch wegen des Bezugs zu einer fixen Primärkopie keine flexible Anpassung an sich dynamisch ändernde Konsistenzanforderungen. Eine Verallgemeinerung des Snapshot-Konzepts bilden Ansätze die gleichberechtigte Replikate erlauben, wobei Regeln zur Änderungspropagierung zwischen je zwei Replikaten zu spezifizieren sind. Derartige Ansätze haben jedoch den Nachteil, daß sie die Konvergenz der Replikate nicht automatisch sicherstellen können.

Der hier vorgestellte Ansatz ermöglicht eine Vermeidung der Nachteile beider Ansätze. Die Idee besteht darin, die Konsistenzanforderungen für jedes Replikat unabhängig von allen anderen Replikaten zu spezifizieren, indem das Replikat in Beziehung gesetzt wird zu einer virtuellen Primärkopie. Die Konsistenzanforderungen werden entlang einer räumlichen und einer zeitlichen Dimension spezifiziert. In der räumlichen Dimension wird ein maximal tolerierbarer Abstand festgelegt. Dabei ist in erster Linie zwischen konsistenten und schwach konsistenten Replikaten zu unterscheiden. In der zeitlichen Dimension wird festgelegt, wann welcher Abstand einzuhalten ist, wobei ereignisorientierte Anforderungen (Spezifikation von Zeitpunkten) und kontinuierliche Anforderungen (Spezifikation von Zeitintervallen) unterschieden werden. Diese Technik ermöglich die Spezifikation dynamisch veränderlicher Konsistenzanforderungen, wobei sowohl Snapshots, als auch Quasi-Kopien sowie aktuelle Replikate spezifiziert werden können.

Die Sicherstellung der so spezifizierten Konsistenzanforderungen basiert auf dem Konzept der Konsistenzinseln. Die Konsistenzinsel eines logischen Datenobjektes ist eine dynamisch veränderliche Teilmenge der Replikate dieses Objekts. Jedes Mitglied der Konsistenzinsel ist ein konsistentes Replikat, das den aktuellen Wert des zugehörigen logischen Objektes trägt. Die Zusammensetzung der Konsistenzinsel entspricht den jeweils momentanen Konsistenzanforderungen, so daß sich stets nur die Replikate in der Konsistenzinsel befinden, für die auch tatsächlich verlangt wird, daß sie zu diesem Zeitpunkt aktuell sein sollen. Replikate können gemäß der spezifizierten Konsistenzanforderungen in die Konsistenzinsel eintreten oder auch wieder aus der Konsistenzinsel austreten. Der Austritt ist jedoch nur erlaubt, wenn noch mindestens ein Replikat in der Konsistenzinsel verbleibt. Die Konsistenzinsel als Ganzes verhält sich aus Sicht der Replikate außerhalb der Konsistenzinsel wie eine virtuelle Primärkopie. Um dieses Verhalten sicherzustellen sind Mechanismen zum Auffinden der momentanen Konsistenzinsel, sowie erweiterte Sperrprotokolle zur Synchronisation von Konsistenzinselein- und Austritten erforderlich.

Der Vorteil des Konzeptes besteht in erster Linie darin, daß die Zahl der zu synchronisieren den Replikate durch dynamische Anpassung an die momentanen Anwendungsanforderungen erheblich reduziert werden kann. Die Zugriffslokalität wird maximiert, indem an jedem Knoten aufgrund der anwendungsspezifischen Anforderungen Replikate lokal bereitgestellt werden. Dadurch, daß sich die Konsistenzinsel eines Objektes als Ganzes wie eine Primärkopie verhält wird die Konvergenz der Replikate sichergestellt.

3.4 A Simulation Study for Shared Disk Parallel Database Systems (B. Kemme, Univ. Erlangen-Nürnberg)

Shared-disk parallel database systems are one form to exploit parallel architectures for high performance transaction processing. A shared-disk parallel database consists of several processing nodes that are loosely coupled via a communication network and share the common database at the disk level.

In our work, we analyze the performance of the recovery component of a shared-disk database system after a node failure. The recovery component has to lead the database back to a consistent state, while transaction processing can proceed on the surviving nodes. Our recovery schemes base on the ARIES method and depend on the algorithms for concurrency and coherency control used by the system. We analyze systems with a centralized and a distributed global lock manager, and page transfer schemes between nodes that write pages back to disk before the transfer or which send the page directly across the network.

We have developed a comprehensive simulation model to study the costs of recovery and the performance of the system during the recovery process. However, we had to make some basic assumptions in order to be able to compare the different architectures. The recovery time mainly depends on the number of pages that must be read from the disk into the buffer during the recovery phase. During recovery, transactions on other nodes are aborted if they want to access a page of the recovery set. The abort rate and with it the throughput rate is determined by the probability transactions want to access pages of the recovery set and by the number of hot spot pages in the recovery set.

Our results show that using a centralized global lock manager the set of pages that has to be recovered can be determined exactly, and less I/O is necessary in comparison to architectures with distributed lock managers that base their information on checkpoint log records. With page transfer schemes that send pages directly across the network many hot spot pages must be recovered because these pages stay dirty in the buffer for a long time while schemes that write pages back to disk before the transfer reduce the number of dirty hot spot pages. Therefore, the latter schemes have higher throughput rates during recovery compared to the first ones.

3.5 Datenverteilung bei Shared-Disk DBS (T. Stoehr, Univ. Leipzig)

Parallele Datenbanksysteme vom Typ Shared Disk (SD) bieten eine hohe Flexibilität für parallele Transaktions- und Query-Verarbeitung. Neben Inter-Query-Parallelität (Erfüllung hoher Durchsatzanforderungen für OLTP-Lasten) gewinnt die Nutzung von Intra-Query-Parallelität (Verarbeitung komplexer Anfragen, z.B. Decision Support Queries) zunehmend an Bedeutung. Speziell für die benötigten Basis-Operatoren wie Scans läßt sich dabei in SD-Umgebungen Intra-Query-Parallelität nutzen.

Die Datenverteilung sollte mit dem Grad der Verarbeitungsparallelität abgestimmt werden, um den Nutzen der Parallelverarbeitung nicht an E/A-Behinderungen scheitern zu lassen. Gerade für die gleichzeitige Verarbeitung von Mischlasten aus OLTP- Transaktionen und komplexen Anfragen (geeignet ist hier einerseits ein E/A-seitiges 'clustering' der Daten zur Erzielung eines hohen Durchsatzes, andererseits aber 'full declustering' für kurze E/A- und damit Antwortzeiten) ist dieses Abstimmung von großer Wichtigkeit. Die Datenverteilung wird zum einen bestimmt durch die Wahl des Verteilungsgrades, also der Anzahl der Platten, auf denen die zu verarbeitenden Daten (Dateien, Relationen) verteilt werden sollen, zum anderen durch das Verteilungsgranulat, also dem Granulat der Allokation der Daten auf den zur Verfügung stehenden Platten.

Erste Ergebnisse bzgl. des Einflusses des Verteilungsgrades wurden mit Hilfe eines umfangreichen Simulationssystems erzielt. Fokus war das Antwortzeitverhalten der Basisoperationen Relationen-Scan (RS), Clustered Index-Scan (CIS) und Non-Clustered Index-Scan (NCIS) in Abhängigkeit vom Grad der Verarbeitungsparallelität und der Wahl des Verteilungsgrades. Unterstellt wurde dabei jeweils 'full declustering' der zu verarbeitenden Relation auf allen Platten, deren Anzahl variiert wurde.

Der RS zeigte hierbei das grösste Parallelisierungspotential, also nahezu linearen Speedup bei Erhöhung des Parallelitätsgrades der Verarbeitung. Der Grad der Datenverteilung musste dabei im Einbenutzerbetrieb der Anzahl N der verarbeitenden Rechner, im Mehrbenutzerbetrieb einem Mehrfachen dieser Anzahl entsprechen, um neben linearen Antwortzeitverbesserungen auch Forderungen nach linearer Durchsatzsteigerung im Mehrbenutzerbetrieb zu erfüllen. Aufgrund der relativ hohen E/A- Rate zeigte sich auch beim NCIS ein vergleichbares Bild mit allerdings geringer ausfallender Antwortzeitverbesserung und einem ggue. dem RS höher liegenden Verteilungsgrad im Mehrbenutzerbetrieb. Der CIS allerdings bot in unsererem untersuchten Szenario kaum Parallelisierungspotential.

Zukünftige Untersuchungen werden sich mit dem Einfluß des Verteilungsgranulates befassen. Simulationsergebnisse zeigen schon jetzt, daß eine Wahl dieser Grösse, die nicht mit dem Verteilungsgrad und dem Grad der Verarbeitungsparallelität abgestimmt ist, zu Engpäßen auf einzelnen Platten führen kann, was z.B. im Fall der parallelen Verarbeitung von RS ein katastrophales Antwortzeitverhalten nach sich zieht.

Zur Untersuchung dieses Einflußes wird ein analytisches Modell erstellt, welches das Antwortzeitverhalten o.g. Scan-Typen in Abhängigkeit der Größen Parallelitäts- und Verteilungsgrad, Verteilungsgranulat, Lastaufkommen, -mixtur etc. im Einbenutzerbetrieb darstellt. Für den praktisch wichtigen Mehrbenutzerbetrieb sind zwei Ansätze denkbar: zum einen die Ableitung der Verteilungsgrößen aus den Ergebnissen des Einbenutzerbetriebes, zum anderen die Bestimmung des Verteilungsgrades aufgrund von Leistungsvorgaben. So sollen Algorithmen entstehen, die auf der Basis vorgegebener Größen (Hardware-Konfiguration, Lastprofil, ...) eine jeweils optimale Datenallokation für eine SD-Umgebung berechnen können.

Ferner wird die Nutzbarkeit von Disk-Arrays im Zusammenhang mit paralleler Scan-Verarbeitung in einer SD-Umgebung untersucht werden.

3.6 Verteiltes Caching (M. Sinnwell, Univ. Saarbrücken)

Netzwerke mit leistungsfähigen Workstations, auch als NOWs bezeichnet, bieten eine hervorragende Basis für kosteneffiziente und skalierbare verteilte Computeranwendungen. Solche Rechnerverbunde wurden bereits in einer Vielzahl von Projekten benutzt, wobei jedoch der Schwerpunkt auf der Ausnutzung der aggregierten CPU-Leistung lag. In datenintensiven Anwendungen, wie z.B. Datenbanksystemen, liegt der Engpass jedoch häufig bei den Plattenzugriffen. Für den Einsatz von NOWs in datenintensiven Anwendungen ist es daher entscheidend, die aggregierten Speicherressourcen (Hauptspeicher und Festplatte) ebenfalls zu nutzen. Da bei schnellen Netzwerken ein Datenobjekt i. allg. schneller aus dem Puffer eines anderen Knotens gelesen werden kann als von der lokalen Platte, werden wir im folgenden eine Heuristik vorstellen, die die Pufferinhalte der beteiligten Knoten so koordiniert, daß die mittlere Antwortzeit von Anfragen minimiert wird.

Um die Leistungsfähigkeit unseres Verfahrens beurteilen zu können, haben wir zwei einfache Heuristiken implementiert, die jeweils einen Grenzfall bezüglich der Kooperation darstellen. Die erste Heuristik agiert völlig egoistisch auf jedem Knoten in dem Sinne, daß nur lokale Informationen benutzt werden, um den Wert eines Objektes zu berechnen. Wir setzen den Wert eines Objektes proportional zu der Zwischenankunftsrate von lokalen Anfragen, und da diese Information i. allg. nicht verfügbar ist, approximieren wir diese Rate durch den LRU-k Algorithmus von O'Neil und Weikum. Die zweite Heuristik stellt im Gegensatz dazu die globale Kooperation zwischen den Knoten in den Vordergrund, indem sie versucht, die Anzahl der Replikate in den verschiedenen Puffern zu minimieren. Dies erreichen wir durch ein einfaches Protokoll, das die Information, ob ein Objekt mehrfach in den Puffern existiert, mit geringem Overhead im System propagiert. Mittels dieser Information ist es nun möglich bei Pufferersetzungen die Objekte, die mehrfach vorhanden sind, zu bevorzugen.

Die beiden vorgestellten Heuristiken versuchen jeweils, unabhängig von den aktuellen System- und Lastparametern, unterschiedliche Ressourcenauslastungen zu minimieren. Während die egoistische Heuristik versucht, die lokale Cache-Trefferrate zu maximieren, und somit die Netzauslastung reduziert, versucht die altruistische Heuristik die globale Cache-Trefferrate zu maximieren und verringert daduch die Plattenauslastung. Da jedoch die Engpassressource (Netz oder Festplatte) von den System- und den Lastparameter abhängig ist, die sich darüber hinaus auch dynamisch ändern können, kann keine dieser Methoden als universell geeignet angesehen werden.

Unsere Heuristik versucht nun durch approximative Kostenformeln den Nutzen eines Objektes dynamisch zu bestimmen, und so flexibel und für jedes Objekt getrennt zwischen egoistischem und altruistischem Verhalten zu wählen. Den Nutzen eines Objektes bestimmen wir anhand der Kosten, die entstehen würden, wenn wir das Objekt aus dem Puffer entfernt würden und direkt anschließend eine Anfrage auf dieses Objekt erfolgen würde. Die Kostenformeln wurden aus einem detaillierten analytischen Modell abgeleitet, das im Gegensatz zu früheren Ansätzen auch die Auslastung von Ressourcen mit berücksichtigt. Nach einer Reihe von Approximationen, die durchgeführt wurden, um den Overhead gering zu halten, sind nur noch die folgenden Informationen notwendig:

- die lokale Hitze, die wir wiederum durch LRU-k approximieren,
- die Information über den Replikationsgrad, die wir wie bei der altruistischen Heuristik bestimmen,
- die globale Hitze, die wir asynchron und schwellwertgesteuert aktualisieren und
- die Ressourcenauslastung, die wir jeweils auf Knotenebene, und damit skalierbar, mitführen.

In einer Reihe von Simulationen wurden die unterschiedlichen Heuristiken über ein breites Spektrum von System- und Lastparametern miteinander verglichen. Dabei hat sich, trotz des zusätzlichen Overheads, die Überlegenheit unseres neuen kosten-basierten Verfahrens gegenüber den einfachen Heuristiken gezeigt.

4 Anfrageoptimierung und Anfragebearbeitung

4.1 Das EXPLORE Projekt - Anfragebearbeitung und -optimierung in parallelen Datenbanksystemen (J. Obermaier, Humboldt Univ. Berlin)

Eine immer umfangreichere zu verwaltende Datenmenge zusammen mit der zunehmenden Komplexität von Anfragen auf der einen Seite und der steigenden notwendigen Transaktionsraten auf der anderen verlangen sehr leistungsfähige Datenbanksysteme. Parallele Verarbeitung ist ein wesentlicher Bestandteil zur Erfüllung der Performance-Anforderungen moderner Datenbankapplikationen.

In EXPLORE (EXtensible ParalleL Optimization REsearch) wird ein erweiterbarer und adaptiver Optimiereransatz verfolgt: Der Optimierer soll an veränderte oder neue Ausführungsumgebungen einfach angepaßt werden können. Er soll mit neuen logischen Operationen, verschiedenen Algorithmen dafür und mit zusätzlichen Optimierungsverfahren und einer Auswahlstrategie zwischen diesen erweitert werden können. Zwei Aspekte werden in EXPLORE besonders betrachtet: Optimierung als Bestandteil der Abarbeitung, d.h. dynamische Optimierung und Parallelisierungsmöglichkeiten des Optimierungsvorgangs selbst.

In EXPLORE wird von dem 'Konsens' der Shared Nothing Architektur als Grundlage für parallele Datenbanksysteme abgewichen. Motiviert durch Entwicklungstrends bei Parallelrechnern wird in EXPLORE von einer Shared Everything Architektur ausgegangen, bzw. bei noch höheren Leistungsanforderungen von einer hybriden Architektur (SE+SN oder SE+SD).

Zur Zeit werden folgende Teilprojekte bearbeitet: - exSPECt: Modellierung der Anfrageauswertung (log.) - Modellierung der Anfrageauswertung (phys.), z.T. in Kooperation mit M. Spiliopoulou, Wiwi, HUB - SODA: Stochastische Algorithmen zur kostenbasierten Optimierung - Genetische Programmierung zur kostenbasierten Optimierung, in Kooperation mit M. Spiliopoulou, Wiwi, HUB - Parallelisierung der Optimierung - dynamische Optimierung, dynamische Auswertungsstrategien - parallele Auswertungsmaschine mit neuen (Kontroll-)Operatoren - Optimierung im erweiterbaren Datenbanksystemen, in Kooperation mit IBM Almaden Research Center - Entwicklung einer offenen Optimiererarchitektur, in Kooperation mit Sharma Chakravarthy, CIS, UFL

4.2 Effiziente daten-parallele Anfragebearbeitung auf Shared-Everything Parallel-Rechnern (S. Manegold, Humboldt Univ. Berlin)

Die Arbeit stellt eine neue Auswertungsstrategie für Pipelining-Segmente auf Shared-Everything Parallel-Rechnern vor. Pipelining-Segmente sind lineare Segmente eines Anfragebaumes, die – mit Ausnahme des letzten Operators – keine blockierenden Operatoren enthalten.

Es wird davon ausgegangen, daß ein Optimierer den Anfragebaum soweit partitioniert hat, daß die Auswertung jedes Pipelining-Segments im Hauptspeicher erfolgen kann, also insbesondere alle Hash-Tabellen der Joins in den Hauptspeicher passen. Die Auswertung erfolgt in zwei Phasen: Zunächst werden alle Hash-Tabellen eines Segments aufgebaut (Build Phase), danach werden die eigentlichen Joins parallel ausgeführt (Probe Phase). Hier wird zunächst nur die zweite Phase betrachtet.

Die hier vorgestellte daten-parallele Auswertungsstrategie (Data Threaded Execution, DTE) geht von der klassischen Inter- und Intra-Operator-Parallelität zur Daten-Parallelität über. Statt jedem Prozessor genau einen (Teil-) Operator zuzuordnen, wird jedem Prozessor ein Thread zugeordnet, der alle Operatoren ausführen kann. Die Eingabe-Tupel des Pipelining-Segments werden in einer globalen Queue zur Verfügung gestellt und "first come first served" auf die Threads verteilt. Ein Tupel verlässt "seinen" Thread erst, wenn es eine Join- oder Selektions-Bedingung nicht erfüllt, oder wenn es das Ende des Pipelining-Segments erreicht hat. Jeder Thread kann das nächste Tupel aus der globalen Queue bearbeiten, sobald er die Bearbeitung eines Tupels beendet hat.

DTE erzielt folgende Vorteile gegenüber klassischem Pipelining: - Es gibt keine Datenabhängigkeiten zwischen den einzelnen Threads, so daß weder startup noch shutdown delay auftreten. - Daten-Parallelität vereinigt Inter- und Intra-Operator-Parallelität. - Da keine statische Zuordnung von Prozessoren zu Operatoren stattfindet, können keine Diskretisierungsfehler auftreten. - Dynamische Lastbalancierung und damit effiziente Ressourcen-Ausnutzung sind implizit. - Hierdurch ist die Resistenz von DTE gegenüber data skew gewährleistet. - Fehler bei der Kostenabschätzung zur Übersetzungszeit haben keinen Einfluss auf die Ausführung, da die Prozessoren nicht entsprechend der Kosten den einzelnen Operatoren zugeordnet werden.

Vergleich von DTE und klassischem Pipelining mit Hilfe von Simulationen haben gezeigt, daß DTE effizientere Ressourcen-Ausnutzung gewährleistet und somit kürzere Ausführungszeiten und einen fast linearen speedup erzielt.

4.3 Modellierung der Anfragebearbeitung in parallel Datenbanksystemen mittels stromverarbeitender Operationen (F. Waas, Humboldt Univ. Berlin)

Parallel Datenbanksysteme erfordern verschiedene Erweiterungen hinsichtlich der Anfragebearbeitung. Im Gegensatz zur sequentiellen Anfragebearbeitung erreicht eine parallele Bearbeitung weitaus höhere Komplexität, die nicht nur aus den Möglichkeiten paralleler Architekturen resultiert, sondern auch aus der gesteigerten Bedeutung dynamischer Aspekte. Die bis dato (auch für sequentielle Systeme) vorgeschlagenen Methoden, insbesondere der Anfragerepräsentation, ermöglichen leider nur begrenzt die gewünschte Berücksichtigung.

Die vorgestellte Arbeit präsentiert einen ersten Ansatz zu einer umfassenden Methodologie, die einerseits einen hohen Abstraktionsgrad, andererseits eine möglichst detailgetreue Wiedergabe der tatsächlichen Implementierung anstrebt.

Die Grundlage der Arbeit bildet das aus der funktionalen Programmierung bekannte Prinzip der Stromverarbeitung. Dabei wird die Anfrageauswertung zunächst als eine Menge beliebiger, über Ströme kommunizierender Systeme, sog. stromverarbeitende Operationen, spezifiziert. Das jeweilige I/O-Verhalten wird durch entsprechende Prädikate definiert. Durch Verfeinerung dieser Operationseinheiten, d.h. durch das Ersetzen einer Operation durch ein System von Operationen einer tieferen Schicht, kann eine Auswertung auf allen relevanten Ebenen mit ein und demselben Formalismus dargestellt werden. Die Anzahl und Bedeutung dieser Ebenen variiert in Abhängigkeit der verwendeten Architektur und erstreckt sich von der deklarativen Beschreibung bis hin zum ausführbaren Programm.

Für die Anfrageoptimierung bedeutet dies, daß der Optimierungsvorgang nicht mehr einem strengen Mehrphasenmodell unterliegen muß und somit kritische Optimierungsschritte reversibel sind. Ferner bietet der Verfeinerungsmechanismus die Möglichkeit unvollständige Spezifikationen zu erstellen, d.h. daß z.B. das Scheduling nur auf Maschinenebene, nicht aber auf Prozessorenebene festgelegt wird und die Auswertungsmaschine selber, zur Laufzeit, die Prozessorzuteilung vornimmt (vgl. DTE, DTE/R).

Herkömmliche Auswertungsparadigmen werden den Beschreibungsmöglichkeiten dieser neuen Methode nicht gerecht. Daher ist eine Verallgemeinerung dieser Paradigmen nötig. Wie anhand von Beispielen verdeutlicht wird, kann dadurch nicht nur eine bessere Modularisierung und Vereinheitlichung erreicht werden, sondern, gerade im Bezug auf asynchrone Kommunikation, eine bedeutende Steigerung der Effizienz.

4.4 Parallelisierte Anfrageoptimierung für parallele Datenbanksysteme (H. Kosch, Lyon)

Die Komplexität der Operationen und die Fülle der zu verarbeitenden Daten lassen den Parallelismus als vielversprechende Zukunftstechnologie für relationale Datenbanksysteme erscheinen. Schon früh profitierten industrielle und wissenschaftliche Forschungseinrichtungen von der aufkommenden Verfügbarkeit von Parallelrechnern. In den letzten sechs Jahren wurden immer neuere, aber auch immer komplexere Techniken entwickelt um die parallele Umgebung besser zu nutzen. Dabei gewann aber die parallele Abfrageoptimierung (PAO), unter Berücksichtigung der Hardwarekomponenten des Systems, immer mehr an Bedeutung.

Neben den klassischen Optimierungszielen, d.h. Operatorenordnung, Zugriffsstrategien und Implementierungsalgorithmen, müssen folgende für den Parallelismus entscheidende Grössen bestimmt werden:

- Parallelitätsgrad einer Operation.
- der Operationen, die parallel ausgeführt werden.
- Anzahl der Operationen, die Daten per Piplineparallelismus konsommieren.
- Synchronisationsbarrieren.

unter der Randbedingung, daß nur ein begrenzter Hauptspeicherbereich für diese Operationen zur Verfügung steht. Der Optimierungprozess wurde deswegen nachhaltig verlangsamt und die Wahrscheinlichkeit, einen guten Ablaufplan zu erzeugen, verringert.

Unser Lösungsvorschlag beruht in der Verwendung von stochastischen Suchstrategien. Beginnend mit einen oder mehreren Initalablaufplaenen werden Transformationen auf die Struktur dieses Ablaufplanes angewandt, bis ein kostengünstiger Plan gefunden wird. Trotz der Vielseitigkeit dieser Optimierungsstrategie müssen hohe Optimierungskosten in Kauf genommen werden, bis ein guter Plan gefunden ist.

Ausgehend von einer grundlegenden Studie der anzuwendenden Transformationen, wurden Gruppen von Transformationen gebildet, die für jede Operationskonstellation anwendbar sind. Darüberhinaus entkoppelten wir die Bestimmung der Zugriffs- und Implementationsstrategien von der eigentlichen Suchstrategie um ein flexibles und erweiterbares Optimierungssystem zu erhalten. Hauptspeicher- und Prozessorenzuweisung erfolgt in einer Phase, wobei Prozessoren von einer initialen Kandidatenmenge entfernt werden, die nicht genügend Hauptspeicher zur Verfügung haben um die Operation auszuführen. Dann wird die Kandidatenmenge möglicher Datenlokalität angepaßt.

Um den Optimierungsprozess noch weiter zu beschleunigen, wird das Potential der parallelen Maschine bereits zur Optimierung genützt. Die Optimierungsstrategie, basierend auf stochastischen Suchstrategien, eignet sich hervorragend zur Parallelisierung.

In einem ersten Ansatz weisen wir jedem Prozessor, der an der Abfrageoptimierung beteiligt ist einen Teil der Initialablaupläne zu. Diese können dann unabhängig voneinander optimiert werden. Dieser Parallelismus, ohne Interferenzen, erreicht optimalen Speed-up.

In einem zweiten Ansatz führen wir den Parallelismus auf der Ebene der Abfrage selber ein. Dieser Ansatz ist motiviert durch die exponentielle Komplexität des Abfrageprozesses in Abhängigkeit von der Anzahl der Operationen. Jeweils Paare von Teilabfragen werden im parallelen optimiert. Die optimalen Teilabfragepläne werden zusammengefügt und nachoptimiert.

Der beschriebene Optimierer ist für SPJ-Abfragen implementiert und umfangreiche Versuchsreihen wurden durchgeführt. Im allgemeinen untersuchten wir die Komplexität des Abfrageprozesses und die Qualität der erzeugten Pläne.

Drei der vier Suchstrategien fanden Pläne, die sich in der Bandbreite guter bis optimaler Kosten bewegten. Die Optimierungszeit der parallelisierten PAO konnte bei geeigneter Wahl des Parallelitaäsgrades stets unter 2s auf einer Sun Sparc 5 gehalten werden und dies für alle getesteten Abfragen (bis zu 11-Wege Joins).

4.5 Parallele Anfrageverarbeitung in MIDAS (M. Jaedicke, C. Nippl, TU München)

In diesem Beitrag wird der aktuelle Stand der Überlegungen zur parallelen Anfrageverarbeitung in MIDAS, dem Münchner Prototyp eines PRDBMS entwickelt im SFB342, Teilprojekt B2 (Prof. Bayer, Prof. Mitschang), vorgestellt. Die Anfrageverarbeitung wird beim Prototyp mit einer evolutionären Strategie parallelisiert, um den bestehenden TransBase-Code möglichst gut wiederverwenden zu können. Im Hinblick auf die Anfrageverarbeitung besteht das Ziel unserer Forschung darin, Konzepte für eine parallele Anfrageverarbeitung primär in Shared-Disk Architekturen zu untersuchen.

Die Anfrageübersetzung soll (zunächst) in drei Phasen erfolgen. In der ersten Phase wird vom existierenden Optimierer ein optimierter Plan für eine sequentielle Ausführung generiert. Optimierungsziel ist dabei die Minimierung des Gesamtaufwands zur Bearbeitung der Anfrage. Daran anschließen soll sich die Parallelisierung durch eine neue Komponente, den Parallelisierer. Dieser muß dafür sorgen, daß die verschiedenen Parallelisierungsmöglichkeiten effektiv zur Anfrageverarbeitung eingesetzt werden. Hierzu wird ein mehrphasiges heuristisches und kostenbasiertes Vorgehen in seinen Grundzügen vorgestellt. Ergebnis ist ein Ausführungsplan, der durch wenige Parameter effektiv an verschiedene Systemzustände anpaßbar sein soll. In der dritten Phase werden diese Parameter unmittelbar vor der interpretativen Ausführung des Plans an den aktuellen Systemzustand angepaßt, um den bestmöglichen Tradeoff zwischen Antwortzeit und Bearbeitungsaufwand zu erzielen. Diese Anpassung wird durch die QEC (Query Execution Control) Komponente erreicht, die die Funktionalitäten Ressourcenallokation, Scheduling und Lastbalancierung integrieren soll. Als Ansatz für das Scheduling ist an eine Spielraumplanung auf der Grundlage des kritischen Pfades der Anfrage gedacht, evtl. verfeinert um eine Steuerung der Prozeßprioritäten.

5 Abschlußdiskussion

Diese Sitzung wurde vorbereitet und moderiert von Prof. Dr. T. Härder. Im ersten Teil der Sitzung wurde versucht, eine grobe und auch nur teilweise Aufarbeitung und Einordnung bisheriger Paralleler DB-Technologie zu geben. Dabei wurde festgestellt, daß insbesondere die Punkte

- Transparenz der Parallelisierung für die Anwendung
- Zerlegbarkeit der Datenobjekte
- Granulate der parallelen Verarbeitung und deren Verteilbarkeit

für die Eignung von Datenmodellen und Mehrrechner-Architekturen für eine parallele DB-Verarbeitung von grundlegender Wichtigkeit sind.

Hieraus ergibt sich - durchaus nicht überraschend -, daß relationale Datenbanksysteme besonders für Parallelverarbeitung geeignet erscheinen. Zum einen liegt das an der vorhandenen deskriptiven Anfragesprache (beispielsweise SQL) und deren inhärente mengenorientierte Verarbeitung. Zum anderen ist eine (horizontale bzw. auch vertikale) Partitionierung flacher Relationen recht einfach zu bewerkstelligen. Insgesamt ergeben sich damit vielfältige Möglichkeiten zur Unterstützung massiver Daten- und Verarbeitungsparallelität.

Die nachstehende Tabelle zeigt den gleichen Sachverhalt auf eine etwas andere Art und Weise. Hier wurde versucht darzustellen, wie sich Forschungs- und Entwicklungsaufwände aufteilen, einmal hinsichtlich der Dimension Datenmodelle und zum anderen hinsichtlich der Dimension Architekturklasse. Bezogen auf die zuletzt genannte Dimension wurden die klassischen Bereiche Shared-Nothing, Shared-Disk, Shared-Memory sowie Mischformen davon unterschieden. Im Bereich Datenmodelle war es uns wichtig Relationenmodell, objektorientierte bzw. Komplexobjekt-Datenmodelle und das Netzwerkmodell (als Vertreter einer etwas älteren Technologie) zu unterscheiden. Die hier eingetragenen Prozentzahlen (nur Schätzwerte) geben zum Ausdruck, daß der Großteil des (bisherigen) Interesses an Paralleler Technologie eindeutig im Bereich relationaler Systeme und dort insbesondere auf dem Gebiet von Shared-Nothing-Architekturen lag bzw. immer noch liegt. Für das Netzwerkmodell (und in gleicher Weise auch für das Hierarchiemodell) gab es keine nennenswerten Anstrengungen, und die Größe des Bereiches Hybride Systemarchitekturen ist momentan noch recht unbekannt.

Unterstützung von Verarbeitungs- und E/A-Parallelität			
Datenmodell Architekturklasse	**Relationenmodell**	**objektorientierte/ Komplexobjekt-Datenmodelle**	**Netzwerkmodell**
Shared-Nothing	80 %	1 %	0 %
Shared-Disc	5 %	2 %	0 %
Shared-Memory	5 %	5 %	0 %
Mischformen	? %	? %	0 %

Im zweiten Teil der Sitzung wurde versucht, einen Ausblick zu erarbeiten. Dabei wurden natürlich alle bisherigen Diskussionen und Ergebnisse mit einbezogen. Die zusammengefaßten Ergebnisse umfaßten insbesondere die folgenden Aspekte:

- Misch-Architekturen
  Einerseits durch Hardware-Entwicklungen und andererseits durch die Evolution der Systeme selbst werden hybride Architekturformen interessant. Hier gilt es nun, die Technologie (Verarbeitungs- und E/A-Parallelität) für hybride Architekturen fortzuschreiben.

- Parallele Objektrelationale Datenbanksysteme
  Mit dem Aufkommen Objektrelationaler Datenbanksysteme ist es nun möglich, Parallelisierungstechnologie auch für objektorientierte bzw. komplexobjektbasierte Datenbanksysteme zu betrachten und insbesondere deren Integration mit relationaler Technologie zu untersuchen.

- Parallele Föderierte Datenbanksysteme
  Aufgrund fortschreitender Vernetzung und Globalisierung kommen Föderierte Datenbanksysteme (etwa als Multi-Server-Systeme) immer häufiger zum Einsatz. Hier gilt es, wie auch schon zuvor erwähnt, bekannte Parallelisierungstechnologie auf diesen Anwendungsfall anzupassen bzw. zu erweitern.

Zum Abschluß der Diskussionen wurde ein weiterer Punkt von allgemeinem Interesse zur Sprache gebracht. Hierbei handelt es sich um die Frage: Wie kann man Parallele Datenbanktechnologie in die diversen und komplexen Anwendungen exportiert werden? Ausgangspunkt zu dieser Fragestellung war die Erfahrung, daß allein der (für die Anwendung transparente) Wechsel von einem sequentiellen zu einem parallelen Datenbanksystem nicht alle Parallelisierungs- und Optimierungsmöglichkeiten ausschöpft. Eine ergänzende und teilweise detailliertere Diskussion zum aktuellen Stand und zukünftigen Trends im Bereich Paralleler Datenbanksysteme (insbesondere zur Thematik der Anfrageoptimierung) findet der interessierte Leser in einem kürzlich im ACM SIGMOD RECORD erschienen Aufsatz von Hasan, Florescu und Valduriez [HFV96].

Literatur:

[**HFV96**] Hasan W., Florescu D. und Valduriez P.: Open Issues in Parallel Query Optimization, in: ACM SIGMOD Record, Vol.25, No.3, September 1996, pp. 28-33

6 Teilnehmerliste

R. Bayer

Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: `bayer@Informatik.TU-Muenchen.DE`

G. Bozas

Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: `bozas@Informatik.TU-Muenchen.DE`

M. Exbrayat

Laboratoire d'Informatique du Parallelisme.
Ecole Normale Superieure de Lyon
F - 69364 Lyon Cedex 07.
e-mail: `exbrayat@lisiflory.insa-lyon.fr`

A. Frank

Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: `franka@Informatik.TU-Muenchen.DE`

J.C. Freytag

Institut für Informatik
Humboldt-Universität zu Berlin
Unter den Linden 6
D-10099 Berlin
e-mail: `freytag@informatik.hu-berlin.de`
http: `//www.dbis.informatik.hu-berlin.de/~explore/exSPECt.html`

M. Gesmann

Department of Computer Science
University of Kaiserslautern
P.O.Box 3049
67653 Kaiserslautern - Germany
e-mail: `gesmann@informatik.uni-kl.de`

T. Härder

Department of Computer Science
University of Kaiserslautern
P.O.Box 3049
67653 Kaiserslautern - Germany
e-mail: `haerder@informatik.uni-kl.de`

M. Jaedicke

Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: `jaedicke@Informatik.TU-Muenchen.DE`

B. Kemme
Universität Erlangen-Nürnberg
Martensstrasse 3A
91058 Erlangen
e-mail: kemme@informatik.uni-erlangen.de

H. Kosch
Laboratoire d'Informatique du Parallelisme.
Ecole Normale Superieure de Lyon
F - 69364 Lyon Cedex 07.
e-mail: Harald.Kosch@lip.ens-lyon.fr

R. Lenz
Universität Erlangen-Nürnberg
Martensstrasse 3A
91058 Erlangen
e-mail: lenz@informatik.uni-erlangen.de

A. Listl
Digital Equipment Gmbh
Gutenbergstraße 3
85774 Unterföhring
e-mail: listl@mfr.dec.com

S. Manegold
Institut für Informatik
Humboldt-Universität zu Berlin
Unter den Linden 6
D-10099 Berlin
e-mail: manegold@dbis.informatik.tu-berlin.de
http: //WWW.DBIS.INFORMATIK.HU-BERLIN.DE/~explore/exSPECt.html

B. Mitschang
Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: mitsch@Informatik.TU-Muenchen.DE

C. Nippl
Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: Clara.Nippl@Informatik.TU-Muenchen.DE

J. Obermaier
Institut für Informatik
Humboldt-Universität zu Berlin
Unter den Linden 6
D-10099 Berlin
e-mail: obermaier@dbis.informatik.tu-berlin.de
http: //www.dbis.informatik.hu-berlin.de/~explore/exSPECt.html

E. Rahm

Department of Computer Science
The University of Leipzig
Augustusplatz 10/11
04109 Leipzig - Germany
e-mail: `rahm@informatik.uni-leipzig.de`

A. Reiser

Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: `reiser@Informatik.TU-Muenchen.DE`

F. Rezende

Department of Computer Science
University of Kaiserslautern
P.O.Box 3049
67653 Kaiserslautern - Germany
e-mail: `rezende@informatik.uni-kl.de`

B. Schiemann

Siemens AG
Dept. ZFE ST SN 1
Otto-Hahn-Ring 6
81739 München
e-mail: `Bjoern.Schiemann@zfe.siemens.de`

H. Schuster

Universität Erlangen-Nürnberg
Martensstrasse 3A
91058 Erlangen
e-mail: `schuster@informatik.uni-erlangen.de`

M. Sinnwell

Fachbereich Informatik
Universität des Saarlandes
Postfach 151150
D-66041 Saarbrücken
e-mail: `sinn@cs.uni-sb.de`

T. Stoehr

Department of Computer Science
The University of Leipzig
Augustusplatz 10/11
04109 Leipzig - Germany
e-mail: `stoehr@informatik.uni-leipzig.de`

K. Unterauer

SNI AG
Dept. OEC BS2000 TP2
Otto-Hahn-Ring 6
81739 München
e-mail: `Karl-Unterauer@mch.sni.de`

F. Waas

Institut für Informatik
Humboldt-Universität zu Berlin
Unter den Linden 6
D-10099 Berlin
e-mail: `waas@dbis.informatik.tu-berlin.de`
http: `//www.dbis.informatik.hu-berlin.de/~explore/exSPECt.html`

J. Weissenfels

Fachbereich 14 - Informatik
Universitaet des Saarlandes
Postfach 151150
D-66041 Saarbruecken
e-mail: `weissenfels@cs.uni-sb.de`

S. Zimmermann

Fakultät für Informatik
Technische Universität München
Orleansstraße 34
D-81667 München
e-mail: `Stephan.Zimmermann@Informatik.TU-Muenchen.DE`