Datenbankreplikation unter Verwendung von Gruppenkommunikation

Bettina Kemme

ETH Zürich
Institut für Informationssysteme
ETH Zentrum, CH-8092 Zürich
E-mail: kemme@inf.ethz.ch
Tel.: ++41-1-632-7258 Fax: ++41-1-632-1172

1 Einführung

Datenreplikation wird als eine elegante Möglichkeit betrachtet, die Probleme, die mit großen verteilten Systemen verbunden sind, zu bewältigen. Replikation erhöht Leistung und Verfügbarkeit, da benötigte Daten als lokale Replikate vorliegen und bei Ausfall einer Kopie auf andere Kopien zurückgegriffen werden kann.

In den letzten 15 Jahren wurden zahlreiche fehlertolerante Protokolle entwickelt, die Zugriffe auf Replikate kontrollieren und die Konsistenz der Replikate gewährleisten [BHG87]. Sie sind eng mit der *Nebenäufigkeitskontrolle* der Datenbanksysteme gekoppelt. Ein Großteil dieser Protokolle unterstützt die *synchrone* Datenreplikation, bei der zu jedem Zeitpunkt nur ein einziger konsistenter Zustand eines Datenobjekts sichtbar ist. Die Replikation von Daten ist also transparent für die Benutzer. Am bekanntesten sind Protokolle, bei denen Leseoperationen rein lokal vorgenommen werden, während Modifikationen an alle verfügbaren Replikate propagiert werden (*All-available-Approach*), und Quorumprotokolle, bei denen sowohl für Schreib- als auch für Leseoperationen auf ein Quorum von Replikaten zugegriffen werden muß. Es wurde eine fundierte Replikationstheorie entwickelt, die es ermöglicht, Korrektheit, Transparenz und Zuverlässigkeit von Protokollen zu überprüfen.

Synchrone Replikation wird jedoch in kommerziellen Datenbanksystemen aus Leistungsgründen so gut wie nicht eingesetzt. Durch den hohen Kommunikationsaufwand erhöhen sich die Antwortzeiten der Transaktionen und damit die Konfliktrate, da die Transaktionen die Objekte sehr lange sperren. Durch die hohe Netzlast sind die Protokolle kaum skalierbar. Schließlich tritt das Phänomen von Deadlocks, die sich auf ein einziges Objekt beziehen, auf [GHOS96]. Eine Ursache dieser Ineffizienz sind die Annahmen, die über die Kommunikationsprimitive gemacht werden. Sie berücksichtigen nicht die Möglichkeiten heutiger Kommunikationsmodule.

Im Bereich der verteilten Systeme sind unter dem Begriff *Gruppenkommunikation* effiziente, fehlertolerante Multicast-Protokolle entwickelt worden, die die physikalische Broadcast-Eigenschaft des Kommunikationsmediums ausnützen. Sie ermöglichen die Koordination von Prozessen in einer verteilten Umgebung. Grundidee ist, daß jeder Prozeß Nachrichten versenden kann, die von allen verfügbaren Prozessen der Gruppe empfangen werden. Es existieren bereits mehrere Forschungsprototypen und auch kommerzielle Systeme, die Gruppenkommunikation unterstützen [BSS91, CT91, MMSA+96, DM96]. Die bereitgestellten Kommunikationsprimitive bieten unterschiedliche Stufen von Fehlertoleranz, Atomarität und der Reihenfolge, in der sie empfangene Nachrichten an die Anwendungsschicht weiterreichen.

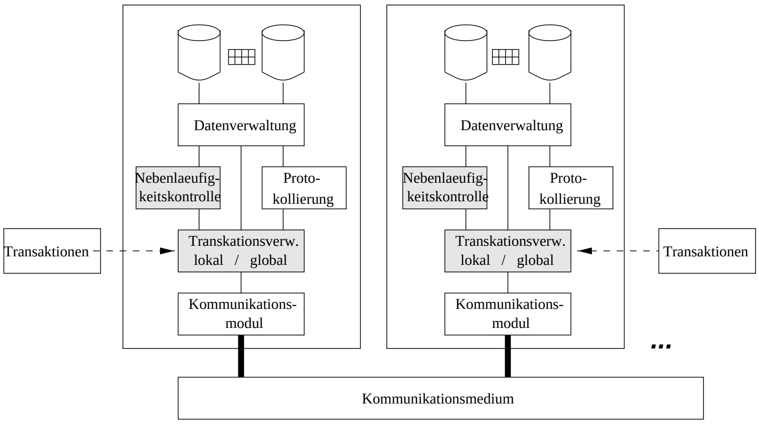

Abbildung 1: Systemarchitektur eines replizierten Datenbanksystems

Unser Ziel ist es, eine Datenbankarchitektur zu bilden, die die Ideen der Gruppenkommunikation in die Datenbankwelt überträgt, um eine synchrone Datenreplikation effizient zu ermöglichen. Die zugrundeliegenden Modelle sind jedoch in mehrfacher Sicht unterschiedlich und müssen miteinander verbunden werden. Beziehen sich Atomarität, Zuverlässigkeit und Reihenfolge in der Gruppenkommunikation auf Nachrichten, so ist die betrachtete Einheit in einem Datenbanksystem eine aus mehreren Operationen bestehende Transaktion, an die die Anforderungen Atomarität, Dauerhaftigkeit und Isolation gestellt werden.

Der Rest dieses Artikels ist wie folgt untergliedert. Im nächsten Abschnitt zeigen wir die Grobarchitektur eines replizierten Datenbanksystems, das auf Gruppenkommunikation aufbaut. Danach beschreiben wir exemplarisch ein Replikationsprotokoll, das synchrone Datenreplikation ermöglicht. Zum Schluß stellen wir weitere Forschungsrichtungen vor.

2 Ein repliziertes Datenbanksystem

Die Systemarchitektur ist in Bild 1 aufgezeigt. Das replizierte Datenbanksystem besteht aus einer Menge von Knoten, die über ein Netzwerk miteinander verbunden sind. Auf jedem Knoten ist eine Instanz des Datenbankverwaltungssystems installiert, das in die üblichen Komponenten wie Transaktionsverwaltung, Nebenäufigkeitskontrolle, Protokollierung, Datenverwaltung und die eigentliche Datenbank unterteilt ist. Wir gehen in einem ersten Schritt von einem vollständig replizierten System aus, d.h. jeder Knoten speichert die gesamte Datenbank. Das Kommunikationsmodul stellt die geeigneten Kommunikationsprimitive zur Verfügung.

2.1 Das Kommunikationsmodul

Unser Kommunikationsmodul benutzt bis jetzt nur Dienstleistungen, wie sie existierende Gruppenkommunikationssyteme anbieten. Wir können dabei zwei Typen von Dienstleistungen unterscheiden. Zum einen nützen wir die Fehlertoleranz. Im Falle von Rechnerausfällen und teilweise auch Netzpartitionen bleibt die Gruppe arbeitsfähig. Desweiteren benützen wir bis jetzt zwei unterschiedliche Kommunikationsprimitive. *Reliable Broadcast* garantiert ein zuverlässiges Senden und Empfangen von Nachrichten. Sendet ein korrekter, nicht ausgefallener Knoten eine Nachricht, dann soll sie von allen verfügbaren Knoten empfangen werden. Weiterhin soll eine gesendete Nachricht von einem Gruppenmitglied nur dann empfangen werden, wenn sie von allen verfügbaren Knoten einer Gruppe empfangen wird. Mit *Atomic Broadcast* werden die Nachrichten zusätzlich von allen Knoten in der selben Reihenfolge empfangen.

2.2 Die Transaktionsverwaltung

In unserem ersten Architekturentwurf sind nur die Transaktionsverwaltung und die Nebenäufigkeitskontrolle für die Konsistenz der replizierten Daten verantwortlich. Für das Datenmanagement und die Protokollierung ist die Replikation transparent und sie arbeiten wie in einem zentralisierten System. Jeder Knoten nimmt Anfragen zur Ausführung von Transaktionen von nahegelegenen Klienten entgegen. Dies sind die lokalen Transaktionen des Knotens. Modifikationen werden auf allen verfügbaren Knoten ausgeführt (*All-available-Approach*). Deshalb werden verändernde Transaktionen den übrigen Knoten bekannt gemacht und dort als globale Transaktionen registriert. Die Nebenläufigkeitskontrolle muß garantieren, daß konfligierende globale Transaktionen an allen Knoten in derselben From serialisiert werden. Um eine akzeptable Leistung zu erreichen, stellen wir folgende Anforderungen:

- Rein lesende Operationen sollen nur lokal ausgeführt werden. Das vermindert nicht nur die Rechenleistung, die an den einzelnen Knoten notwendig ist, sondern auch die Anzahl bzw. Größe der Nachrichten, da nur Modifikationen an die übrigen Knoten propagiert werden.
- Die Anzahl der Nachrichten sollte so niedrig wie möglich sein. Deshalb werden Schreiboperationen bis zum Ende der Transaktion verzögert, um sie mit einer Nachricht an alle Knoten zu versenden.

Im folgenden skizzieren wir ein Protokoll, das die oben genannten Anforderungen erfüllt [AAAS96]. Es benötigt zwei Nachrichten pro Transaktion und garantiert volle Serialisierbarkeit. Es benutzt ein striktes 2-Phasen-Sperrprotokoll [BHG87].

- Für eine Leseoperation wird lokal eine Sperre angefordert.
- Schreiboperationen werden am Ende der Transaktion in einer Nachricht mit *Atomic Broadcast* an die übrigen Knoten gesandt. Dadurch werden alle Transaktionen durch die Reihenfolge, in der ihre Schreiboperationen empfangen werden, serialisiert.
- Wird eine Nachricht mit Schreiboperation empfangen, versucht die Sperrverwaltung für alle betroffenen Objekte Sperren anzufordern. Gibt es bereits eine Schreibsperre auf einem Objekt, wird gewartet, bis die Sperre freigegeben wird. Für jede existierende Lesesperre wird überprüft, ob die lesende Transaktion bereits eine *Commitable*-Nachricht gesendet hat. In diesem Fall wird gewartet, bis die Sperre freigegeben wird. Andernfalls wird die lesende Transaktion abgebrochen. Sollte Sie bereits eine Nachricht mit ihren Schreiboperationen versendet haben, versendet sie eine *Abort*-Nachricht.
- Empfängt ein Knoten eine Nachricht mit den Schreiboperationen einer lokalen Transaktion, kann diese aus konfliktbedingten Gründen nicht mehr abgebrochen werden. Da alle Schreibsperren auf einmal und am Ende der Transaktion angefordert werden und Schreiboperationen Priorität vor Leseoperationen haben, können keine Deadlocks entstehen. Deshalb kann der Knoten zu diesem Zeitpunkt die *Commitable*-Nachricht versenden.
- *Commitable*- und *Abort*-Nachrichten werden mit dem *Reliable Broadcast* versendet.

Da Leseoperationen nur lokal bekannt sind, müssen Schreiboperationen Priorität vor Leseoperationen gegeben werden. Das kann zu unnötigem Abbruch von Transaktionen führen, vermeidet aber, daß verteilte Deadlocks entstehen können.

Ausgehend von diesem Protokoll können nun Optimierungen durchgeführt werden. In kommerziellen Datenbanksystemen werden vielfach Techniken wie *Field Calls, Cursor Stability* und *Snapshot Isolation* [GR93, BBG+95] eingesetzt, um die Konfliktrate herabzusetzen. Sie lassen sich jedoch entweder nur für bestimmte Arten von Operationen und Transaktionen anwenden oder garantieren keine Serialisierbarkeit mehr. Die meisten dieser Protokolle können auf einfache Weise auf replizierten Systeme übertragen werden, ohne den Nachrichtenaufwand zu erhöhen [Kem97]. Wir haben z.B. ein Protokoll entwickelt, das den Bedingungen der Snapshot Isolation genügt und nur eine Nachricht pro Transaktion benötigt.

2.3 Weitere Forschungsrichtungen

Die Weiterentwicklung dieser Architektur und der präsentierten Ideen erfolgt in zwei Richtungen. Die erste ist praxisorientiert und hat den Entwurf und die Implementierung eines Prototypen zum Ziel. Momentan entwickeln wir ein Simulationstool, mit dem die Einsetzbarkeit von Nebenläufigkeits- und Kommunikationsprotokollen unter verschiedenen Konfigurationen und Transaktionslasten getestet werden kann. Die Ergebnisse werden dann in die Realisierung des Prototypen einfließen. Dieser soll existierende Datenbanksysteme verwenden. Deshalb müssen wir die Replikationskontrolle als Schicht über dem Datenbanksystem ansiedeln. Der Prototyp wird dazu dienen, Probleme aufzudecken, unsere theoretischen Ansätze zu verfizieren und die Skalierbarkeit der Datenreplikation zu untersuchen.

Im theoretischen Teil unserer Forschungsarbeit behandeln wir folgende Probleme:

- *Fehlertoleranz*:
  In Gruppenkommunikationssystemen einigen sich bei Ausfall eines Knotens die übrigen Knoten auf eine neue Gruppenkonfiguration. Dazu werden zusätzliche Nachrichten versendet. Wir untersuchen, inwieweit dieser Nachrichtenaustausch verwendet werden kann oder erweitert werden muß, um auch Entscheidungen über Transaktionen zu treffen, die von dem ausgefallenen Knoten initiiert worden sind. Will sich ein ausgefallener Knoten wieder bei der Gruppe anmelden, muß ein Datenbanksystem Recovery durchgeführen, um seine Datenbank wieder in einen konsistenten Zustand zurückzuführen.
  Wir haben bis jetzt nur Kommunikationsprimitive verwendet, die eine hohe Fehlertoleranz bieten. Doch diese Protokolle sind auch sehr teuer bezüglich der Anzahl der Nachrichten und der Verzögerung bis zum Empfangen einer Nachricht. Wir untersuchen, ob und inwieweit das Datenbanksystem die Sicherstellung der Fehlertoleranz übernehmen kann, so daß im fehlerfreien Fall der Nachrichtenaustausch schneller erfolgen kann. Hier wird es bestimmt einen Trade-Off bezüglich Effizienz im fehlerfreien Fall und Aufwand und Blockierungsgefahr bei einem Ausfall geben.
- *Partielle Datenreplikation*:
  Ist die Datenbank nur partiell repliziert, kann es schreibende Transaktionen geben, die rein lokal nur auf einem Knoten arbeiten und dem globalen System nicht bekannt sind. Sie können bewirken, daß globale Transaktionen trotz der durch das Kommunikationssystem vorgegebenen totalen Ordnung an verschiedenen Knoten unterschiedlich serialisiert wer-

den [Alo97]. Wir untersuchen, welche Bedingungen die Nebenläufigkeitskontrolle erfüllen muß, um dies zu vermeiden.

- *Hetereogenität und Nichtdeterminismus*:
  In unserem Ansatz muß sich das Datenbanksystem deterministisch verhalten (*finite automaton approach*). Nur der Knoten, der die Transaktion anstößt, entscheidet über Erfolg oder Abbruch der Transaktion, die anderen Knoten müssen sich dieser Entscheidung beugen (finite automaton approach). Dies fordert natürlich nicht nur ein homogenes Datenbanksystem, sondern auch, daß einzelne Transaktionen nicht einfach wegen fehlendem Plattenplatz oder ähnlichem abgebrochen werden können. Es muß untersucht werden, in welcher Form diese Anforderungen abgeschwächt werden können und welche Minimalforderungen bestehen bleiben müssen.
- *Sprachkonstrukte*:
  Es ist notwendig, geeignete Sprachkonstrukte und Modellierungsmöglichkeiten bereitzustellen, um das replizierte Datenbanksystem in effizienter und transparenter Weise zu konfigurieren.

Literatur

[AAAS96] D. Agrawal, G. Alonso, A. El Abbadi, and I. Stanoi. Exploiting atomic broadcast in replicated databases. Technical report, Institut für Informationssysteme, ETH Zürich, October 1996. In Preperation.

[Alo97] G. Alonso. Partial database replication and group communication primitives. In *2nd European Research Seminar on Advances in Distributed Systems (ERSADS'97)*, Zinal (Valais, Switzerland), March 1997.

[BBG+95] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. A critique of ANSI SQL isolation levels. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–10, June 1995.

[BHG87] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison Wesley, Massachusetts, 1987.

[BSS91] K. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.

[CT91] T. D. Chandra and S. Toueg. Unreliable failure detectors for asynchronous systems. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, pages 325–340, August 1991.

[DM96] D. Dolev and D. Malki. The transis approach to high availability cluster communication. *Communications of the ACM*, 39(4):63–70, April 1996.

[GHOS96] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The dangers of replication and a solution. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 173–182, June 1996.

[GR93] J. Gray and A. Reuter. *Transaction Processing*. Morgan Kaufmann, 1993.

[Kem97] B. Kemme. Database replication on top of group communication. Technical report, Institut für Informationssysteme, ETH Zürich, March 1997. In Preperation.

[MMSA+96] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM*, 39(4):54–63, April 1996.