Proc. GI-Workshop "Verteilte Datenbanksysteme und verteilte Datenverwaltung in Forschung und Praxis"
Datenbank-Rundbrief Nr. 8 der GI-Fachgruppe Datenbanken, Nov. 1991, S. 47-56

Klassifikation und Vergleich verteilter Transaktionssysteme

Erhard Rahm

Universität Kaiserslautern, Fachbereich Informatik

E-Mail: rahm@informatik.uni-kl.de

1. Einführung

Transaktionssysteme [11] befinden sich in weit verbreitetem Einsatz in der kommerziellen Datenverarbeitung, insbesondere im Rahmen von Auskunfts- und Buchungssystemen sowie zur aktenlosen Sachbearbeitung, z.B. bei Behörden oder Versicherungen. Kennzeichnende Eigenschaften solcher Systeme sind u.a. die dialogorientierte Verarbeitung von meist einfachen Transaktionen (Bsp.: Kontenbuchung) sowie die Bereitstellung einer hohen (maskenorientierten) Endbenutzer-Schnittstelle. Hauptkomponenten eines Transaktionssystemes sind das Datenbanksystem (DBS), bestehend aus Datenbank und Datenbankverwaltungssystem (DBVS), sowie ein DC-System, bestehend aus einem TP-Monitor sowie der Menge der Transaktions- oder Anwendungsprogramme. Der TP-Monitor kontrolliert die Ausführung der Programme und realisiert die Kommunikation von Programmen mit Terminals sowie mit dem DBVS. Insbesondere können so physische Eigenschaften der Terminals sowie des Verbindungsnetzwerks für den Programmierer transparent gehalten werden. TP-Monitor und DBVS führen ein koordiniertes Commit-Protokoll aus, um die Atomaritäts-Eigenschaft einer Transaktion zu gewährleisten.

Traditionellerweiser erfolgt die Transaktionsverarbeitung zentralisiert auf einem Großrechner (Mainframe). Eine Vielzahl von Gründen führte zu verschiedenen Typen von verteilten Transaktionssystemen, welche in diesem Aufsatz klassifiziert werden sollen. Hauptgründe für den Einsatz verteilter Transaktionssysteme sind:

1. Leistungsanforderungen (Abwicklung hoher Transaktionsraten, kurze Antwortzeiten)
   Für Hochleistungsanwendungen sind die geforderten Transaktionsraten (mehrere Tausend Transaktionen pro Sekunde) durch einen einzigen Rechner nicht zu bewältigen. Die Antwortzeit komplexerer Transaktionen (Anfragen) soll ggf. durch Parallelverarbeitung verringert werden.
   Von genereller Wichtigkeit im verteilten Fall ist, daß möglichst wenig Kommunikationsverzögerungen und -Overhead bei der Abarbeitung einer Transaktion anfallen und daß alle Rechner effektiv genutzt werden können (Lastbalancierung).
2. Hohe Verfügbarkeit
3. Modulare Wachstumsfähigkeit des Systems
4. Anpassung des Transaktionssystems an die Organisationsstruktur
   => geographische Verteilung des Systems und der Datenbank(en)
5. Unterstützung von Knotenautonomie
6. Integrierter Zugriff auf heterogene Datenbanken
7. Verbesserung der Kosteneffektivität durch verstärkten Einsatz von Mikroprozessoren
   (z.B. in Workstations oder dedizierten Backend-Rechnern)

Bei einem Vergleich verschiedener Architekturformen verteilter Transaktionssysteme ist zu prüfen, in welchem Ausmaß sie die genannten Anforderungen erfüllen können. Dabei sind jedoch noch weitere Aspekte zu berücksichtigen, z.B. Einfachheit der Realisierung, Programmierung (Ortstransparenz !) und Administration.

Bei der Realisierung verteilter Transaktionssysteme sind einige grundsätzliche **Allokationsprobleme** zu lösen, insbesondere hinsichtlich der Rechnerzuordnung von Systemfunktionen (TP-Monitor, DBVS), Anwendungsfunktionen (Transaktionsprogrammen), Daten und Lasteinheiten. Bei der Allokation von Systemfunktionen unterscheiden wir zwischen zwei generellen Alternativen, welche die Basis unseres Klassifikationsschema bilden. Bei *horizontal verteilten Transaktionssystemen* liegt eine Replikation von Systemfunktionen vor, so daß jeder Rechner die gleiche

Funktionalität hinsichtlich der Transaktionsbearbeitung aufweist. Dagegen erfolgt bei den *vertikal verteilten Transaktionssystemen* eine Partitionierung der Systemfunktionen auf Front-End- und Back-End-Rechner (Server), welche zu einer funktionalen Spezialisierung der Rechner führt. In beiden Fällen kann die Verteilung auf mehreren Ebenen des DC-Systems oder des DBVS erfolgen, ebenso lassen sich beide Verteilformen kombinieren. Verteilte Datenbanksysteme zählen zur Klasse der horizontal verteilten Transaktionssysteme mit Verteilung im DBVS; Workstation-Server-Systeme und Datenbankmaschinen sind Beispiele vertikal verteilter Transaktionssysteme. Verteilte TP-Monitore können sowohl eine horizontale als eine vertikale Verteilung unterstützen. Eine genauere Betrachtung der verschiedenen Ausprägungen horizontal und vertikal verteilter Transaktionssysteme erfolgt in Kap. 2 und 3.

Bei der Rechnerzuordnung von Transaktionsprogrammen (TAP) und Daten kann ebenfalls zwischen Partitionierung und (teilweiser oder voller) Replikation unterschieden werden. Dabei liegt im Falle der Partitionierung ein Programm/Datum an genau einem Rechner vor, während im Falle der Replikation mehrere oder alle Rechner eine Kopie des Programmes/Datums halten. Replikation erhöht die Möglichkeiten der Lastbalancierung, da dann mehrere Rechner zur Auswahl stehen, um ein bestimmtes Programm bzw. einen DB-Zugriff zu bearbeiten. Auf der anderen Seite entsteht ein erhöhter Speicherplatzbedarf und Änderungsaufwand. Eine weitere Allokationsform ist das sogenannte 'Sharing', bei dem alle Rechner direkt auf alle Platten zugreifen können und damit auf die dort gespeicherten TAP und Daten. Diese Strategie vermeidet, eine Partitionierung von Programmen und Daten vornehmen zu müssen, und bietet dasselbe Optimierungspotential zur Lastbalancierung wie bei voller Replikation. Eine Einschränkung dabei ist, daß die Rechner aufgrund der Plattenanbindung nicht geographisch verteilt werden können.

Bei der Lastzuordnung sind mehrere Verteileinheiten möglich (Transaktionsaufträge, DB-Operationen, ...), welche durch die gewählte Allokation von Funktionen und Daten mitbestimmt werden. Generell sollte es möglich sein, den aktuellen Systemzustand (Rechnerauslastung, etc.) bei der Lastzuordnung zu berücksichtigen, um eine effektive Lastbalancierung zu erzielen.

2. Horizontal verteilte Transaktionssysteme

In horizontal verteilten Transaktionssystemen sind die beteiligten Rechner funktional äquivalent hinsichtlich der Transaktionsverarbeitung. Es liegt eine Replikation von Systemfunktionen vor, so daß in jedem Rechner ein TP-Monitor und ein DBVS laufen und zusammenarbeiten. Allerdings wird keine Homogenität der Rechner gefordert; vielmehr können verschiedene TP-Monitore und DBVS auf den einzelnen Rechnern laufen und heterogene Betriebssysteme und Rechner-Hardware vorliegen.

Eine weitergehende Unterteilung horizontal verteilter Transaktionssysteme ist möglich, wenn unterschieden wird, auf welcher Ebene innerhalb des DC-Systems oder des DBS die Rechner miteinander kooperieren, wo also die Verteilung sichtbar ist. Hierzu unterteilen wir zwischen Verteilung nur im DC-System, Verteilung im DBS (Mehrrechner-DBS) sowie Verteilung im DC-System und im DBS.

Verteilung (nur) im DC-System

In diesem Fall erfolgt die Kommunikation zwischen den TP-Monitoren der Rechner, während die DBVS voneinander unabhängig sind. Dies hat den Vorteil einer relativ geringen Systemkomplexität, da bei den DBVS (nahezu) keine Erweiterung im Vergleich zum zentralisierten System erforderlich ist. Zudem ist es möglich, heterogene DBVS auf verschiedenen Rechnern einzusetzen. Auch wird ein hohes Maß an Knotenautonomie gewährleistet, da jeder Rechner eine unabhängige Datenbank hält, die durch ein eigenes konzeptionelles Schema beschrieben wird.

Aufgrund der Unabhängigkeit der DBVS liegt hierbei in der Regel eine **Partitionierung der Daten** vor (Datenreplikation müßte ansonsten im DC-System gewartet werden; analog müßte bei einem Sharing die notwendige globale Synchronisierung der Datenzugriffe im DC-System erfolgen). Die Einheit der Datenverteilung ist sehr grob (ganze Datenbanken). Nachteile ergeben sich auch hinsichtlich Verfügbarkeitsanforderungen: fällt ein Rechner aus, sind die dort verwalteten Daten nicht mehr erreichbar. (Bei Mehrrechner-DBS dagegen kann nach Durchführung bestimmter Recovery-Aktionen einer der überlebenden Rechner die Daten des ausgefallenen Rechners zeitweilig mitverwalten). Aufgrund der Unabhängigkeit der DBVS können verteilte Deadlocks i.a. nur über einen Timeout-Mechanismus aufgelöst werden.

Nach [10] lassen sich drei Verteilungsformen im DC-System unterscheiden, welche durch verschiedene Verteileinheiten gekennzeichnet sind:

(1) *Transaction Routing*
Hierbei werden ganze Transaktionsaufträge zwischen den Rechnern verteilt. Wird nur diese Verteilform unterstützt, d.h. wenn keine weitere Verteilung auf tieferer Ebene (mit feinerem Verteilgranulat) erfolgt, muß eine Transaktion vollkommen auf einem Rechner ausgeführt werden, und es können somit nur die lokal erreichbaren Daten referenziert werden. Diese Verteilform setzt daher auch eine Partitionierung der TAP voraus, wobei der TP-Monitor die Programmzuordnung kennt und somit einen Auftrag an den Rechner weiterleiten kann, wo das entsprechende TAP vorliegt. Dennoch ist diese Verteilform allein nicht ausreichend, da keine echt verteilte Transaktionsausführung erfolgt.
Transaction Routing wird u.a. durch die TP-Monitore CICS und IMS-DC (MSC-Komponente: Multiple Systems Coupling) von IBM unterstützt.

(2) *Verteilte Programmierung*
In diesem Fall kann auf entfernte Daten durch Aufruf eines Unter- bzw. Teilprogrammes an dem entsprechenden Rechner zugegriffen werden, d.h. die Verteilung erfolgt auf Ebene der TAP. Jedes Teilprogramm kann nur lokale Daten eines Rechners referenzieren, wenn keine weitere Verteilung auf tieferer Ebene erfolgt, so daß auch hier eine Partitionierung von (Teil-)Programmen vorliegt. Orts-transparenz kann i.a. nicht erreicht werden, da für eine verteilte Anwendung die Daten- und Programmpartitionierung so aufeinander abgestimmt werden müssen, daß die genannte Restriktion erfüllt wird. Zudem werden in existierenden DC-Systemen entfernte Unterprogramme (via Remote Procedure Call) zum Teil anders als lokale aufgerufen. Aufgabe der beteiligten TP-Monitore ist die Weiterleitung der entfernten Aufrufe/Antworten sowie die Unterstützung eines verteilten Commit-Protokolles, um die Alles-oder-Nichts-Eigenschaft der Transaktion zu gewährleisten. Die DBVS sind über die TP-Monitore indirekt an einem verteilten Commit beteiligt.
Verteilte Programmierung wird u.a. von UTM (UTM-D) und CICS (Distributed Transaction Processing) unterstützt. Sybase bietet im Prinzip auch diese Verteilungsform, wenngleich dort die TAP als sogenannte 'stored procedures' durch das DBVS verwaltet werden.

(3) *Verteilung von DML-Befehlen (DB-Operationen)*
Hierbei werden DML-Befehle von dem TP-Monitor zwischen den Rechnern ausgetauscht, mit der Einschränkung, daß jede DB-Operation nur Daten eines Rechners referenzieren darf (keine Kooperation zwischen den DBVS). Das feinere Verteilgranulat gestattet mehr Flexibilität für externe Datenzugriffe als bei der verteilten Programmierung, jedoch auf Kosten eines potentiell höheren Kommunikationsaufwandes. Ortstransparenz kann nicht erzielt werden, da bei der Formulierung der DB-Operationen explizit auf das entsprechende DB-Schema Bezug genommen werden muß (der tatsächliche Ort der Datenbanken kann durch Verwendung logischer Namen verborgen werden, nicht jedoch die Unterscheidung mehrerer Datenbanken und ihrer Schemata). Operationen, welche Daten verschiedener Rechner betreffen, müssen explizit ausprogrammiert werden (z.B. Join-Berechnung zwischen Relationen mehrerer Rechner). Wie bei der verteilten Programmierung müssen die beteiligten TP-Monitore ein verteiltes Commit-Protokoll unterstützen.
Die Verteilung von DML-Befehlen wird u.a. von CICS (Function Request Shipping) unterstützt.

Verteilung im DBS (horizontal verteilte Mehrrechner-Datenbanksysteme)

In diesem Fall erfolgt die Kooperation zwischen den DBVS der Rechner. Die DC-Systeme der Rechner können unabhängig voneinander arbeiten und verschiedene TP-Monitore benutzen. Eine grobe Unterteilung von Mehrrechner-DBS ist in Bild 1 gezeigt. Als primäres Unterscheidungsmerkmale dient dabei die Rechnerzuordnung der Externspeicher (Platten), wobei zwischen Partitionierung ('Shared Nothing', SN) und Sharing ('Shared Disk' (SD), DB-Sharing) unterschieden wird. Im Falle von SN ist jede Platte nur einem Rechner zugeordnet; die Daten sind entweder ebenfalls partitioniert oder repliziert (im Falle der Replikation obliegt deren Wartung den DBVS). Bei SD kann jeder Rechner und jedes DBVS direkt auf alle Platten und somit die gesamte Datenbank zugreifen. Zwei wei-

tere Klassifikationsmerkmale erlauben die Unterscheidung zwischen integrierten und föderativen sowie zwischen homogenen und heterogenen Mehrrechner-DBS.

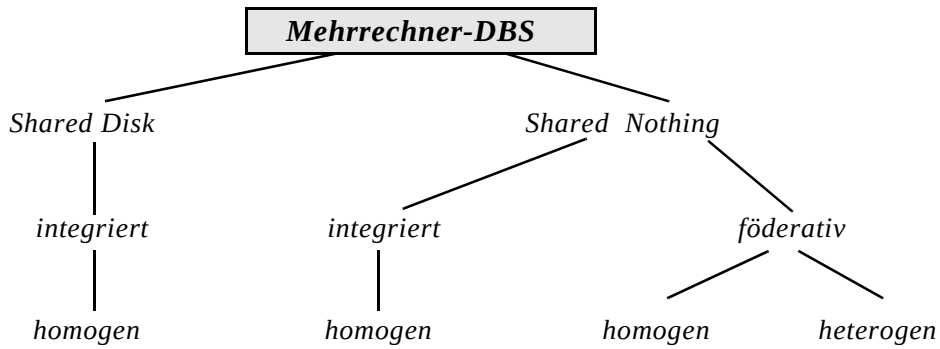


Bild 1: Grobklassifikation horizontal verteilter Mehrrechner-Datenbanksysteme

Integrierte Mehrrechner-DBS sind dadurch gekennzeichnet, daß sich alle Rechner eine gemeinsame Datenbank teilen, die durch ein einziges konzeptionelles Schema beschrieben ist. Da dieses gemeinsame Schema von allen Rechnern unterstützt wird, kann den TAP volle Ortstransparenz geboten werden; sie können auf die Datenbank wie im lokalen Fall zugreifen. Auf der anderen Seite ist die Autonomie der einzelnen Rechner/DBVS stark eingeschränkt, da Schemaänderungen, Vergabe von Zugriffsrechten, etc. global koordiniert werden müssen. Daneben wird vorausgesetzt, daß jede DB-Operation auf allen Rechnern gleichermaßen gestartet werden kann und daß potentiell alle DBVS bei der Abarbeitung einer DB-Operation kooperieren. Diese Randbedingungen erfordern i.d.R., daß die DBVS in allen Rechnern identisch (homogen) sind.

Verteilte Datenbanksysteme sind die bekanntesten Vertreter integrierter Mehrrechner-DBS; sie gehören zur Kategorie 'Shared Nothing'. Bei ihnen kann die Kooperation innerhalb des DBVS auch auf mehreren Ebenen mit unterschiedlichen Verteilgranulaten erfolgen: so können mengenorientierte Teiloperationen, Satz- oder Seitenanforderungen zwischen den Rechnern verschickt werden, um auf externe Daten zuzugreifen. Im Gegensatz zu verteilten DC-Systemen können bei der Datenzuordnung relativ feine Verteileinheiten gewählt werden (z.B. horizontale und vertikale Aufteilung von Relationen). Die beteiligten Rechner können sowohl ortsverteilt als auch lokal verteilt angeordnet sein. Technische Probleme, die bei diesem Ansatz zu lösen sind, beinhalten die Bestimmung der DB-Fragmentierung und DB-Allokation, Erstellung verteilter/paralleler Ausführungspläne, verteilte Commit-Behandlung, globale Deadlock-Behandlung sowie die Aktualisierung replizierter Datenbanken. Liegt eine Partitionierung der Daten vor, so ergeben sich relativ geringe Freiheitsgrade zur dynamischen Lastbalancierung, da die Datenzuordnung relativ statisch ist und jeder Rechner Zugriffe auf seine Partition weitgehend selbst bearbeiten muß [13].

DB-Sharing-Systeme [12] ('Shared Disk') verkörpern eine weitere Klasse integrierter Mehrrechner-DBS. Aufgrund der direkten Plattenanbindung aller Rechner ist hierbei keine physische Datenaufteilung unter den Rechnern vorzunehmen. Insbesondere kann eine DB-Operation auch von jedem Rechner gleichermaßen abgearbeitet werden (keine Erstellung verteilter Ausführungspläne), wodurch eine hohe Flexibilität zur dynamischen Lastbalancierung entsteht. Kommunikation zwischen den DBVS wird v.a. zur globalen Synchronisation der DB-Zugriffe notwendig. Weitere technische Probleme betreffen die Behandlung von Pufferinvalidierungen, Logging/Recovery sowie die Koordination von Lastverteilung und Synchronisation [12]. Existierende DB-Sharing-Systeme sind u.a. IMS Data Sharing, DEC's Datenbanksysteme (RDB, VAX-DBMS) in der Vax-Cluster-Umgebung sowie Oracle V6.2.

Föderative Mehrrechner-DBS streben nach größerer Knotenautonomie im Vergleich zu den integrierten Systemen, wobei die beteiligten DBS entweder homogen oder heterogen sein können. Aufgrund der häufig geforderten geographischen Verteilung des Systems sowie der möglichst weitgehenden Rechnerunabhängigkeit kommt zur Realisierung dieser Systeme nur eine Partitionierung der Externspeicher in Betracht (Shared Nothing). Kennzeichnende Eigenschaft der föderativen DBS ist, daß (ähnlich wie bei den verteilten DC-Systemen) jeder Rechner eine eigene Datenbank verwaltet, die durch ein lokales (privates) konzeptionelles Schema beschrieben ist. Allerdings soll jetzt durch eine begrenzte Kooperation der DBVS es möglich werden, über das DBVS auf bestimmte Daten anderer

Rechner zuzugreifen, falls dies von dem 'Besitzer' der Daten zugelassen wird. Ein Realisierungsansatz hierzu sieht vor, daß jeder Rechner Teile seines konzeptionellen Schemas exportiert und somit externe Zugriffe auf die betreffenden DB-Bereiche zuläßt. Im Idealfall kann Ortstransparenz für die Benutzer (TAP) durch Definition integrierter externer Schemata (Sichten) erreicht werden, wobei das DBVS dann eine automatische Abbildung auf das lokale konzeptionelle Schema sowie die importierten Schemafragmente anderer Rechner vornimmt und für rechnerübergreifende Operationen (z.B. Joins) eine geeignete Anfragezerlegung ausführt. Allerdings ergeben sich bei der Schemaintegration bereits im Fall homogener DBVS (mit gleichem DB-Modell und Anfragesprache) vielfach semantische Abbildungsprobleme, welche nicht automatisch aufgelöst werden können [14]. Aufgrund der Änderungsproblematik auf Sichten wird auch oft nur lesender Zugriff auf externe Daten unterstützt. Weitere technische Probleme betreffen v.a. die Unterstützung heterogener Datenmodelle und Anfragesprachen sowie die Koordinierung unterschiedlicher Methoden zur Transaktionsverwaltung in den beteiligten Rechnern. Ingres/Star repräsentiert ein kommerziell verfügbares föderatives Mehrrechner-DBS [2].

Eine weitergehende Klassifizierung integrierter Mehrrechner-DBS findet sich in [7]. Unterschiedliche Realisierungsformen föderativer DBS werden in [1] gegenübergestellt.

Verteilung im DC-System und im DBS

Die angesprochenen Verteilformen im DC-System und im DBS lassen sich zum Teil kombinieren, wodurch sich i.d.R. größere Freiheitsgrade bezüglich der Last-, Programm- und Datenverteilung ergeben, jedoch auf Kosten einer erhöhten Komplexität. Auch lassen sich einige Nachteile verteilter DC-System beseitigen, wenn zusätzlich ein Mehrrechner-DBS eingesetzt wird (z.B. können durch Verwendung eines Verteilten DBS Ortstransparenz sowie feine Verteileinheiten bei der Datenzuordnung unterstützt werden). Bei Einsatz eines Mehrrechner-DBS erscheint ein zusätzliches Transaction Routing zur Lastverteilung generell sinnvoll, um damit einen Transaktionsauftrag einem Rechner zuzuweisen, der nicht überlastet ist und bei dem eine effiziente Ausführung erwartet werden kann (und auf dem das zugehörige TAP ausführbar ist). Im Zusammenhang mit DB-Sharing erscheinen dagegen die Verteilformen auf Ebene der Teilprogramme und DML-Befehle nicht sinnvoll, da jeder Rechner auf die gesamte Datenbank zugreifen kann.

Tandem unterstützt mit dem TP-Monitor Pathway sowie dem verteilten DBS NonStop SQL eine spezielle Verteilung auf Teilprogrammebene sowie eine verteilte und parallele Anfragebearbeitung im DBVS.

Vergleichende Bewertung

In Bild 2 sind für einige der angesprochenen Architekturformen horizontal verteilter Transaktionssysteme die qualitativen Bewertungen zur Erleichterung des Vergleichs zusammengefaßt. Man erkennt, daß für jede Verteilform Vor- und Nachteile hinsichtlich der angeführten Bewertungskriterien bestehen, so daß es (wie zu erwarten) keinen 'idealen' Typ eines verteilten Transaktionssystemes gibt. So liegen die Vorteile verteilter DC-Systeme in der vergleichsweisen einfachen Realisierbarkeit sowie der Unterstützung einer hohen Knotenautonomie und heterogener Datenbanken, die geographisch verteilt vorliegen können. Die integrierten Mehrrechner-DBS vom Typ SN und SD dagegen erlauben eine einfache Erstellung der Anwendungen (Ortstransparenz) und bieten die größten Freiheitsgrade bezüglich der Last-, Programm- und Datenzuordnung sowie der Nutzung von Parallelität innerhalb einer Transaktion. Diese Ansätze sind daher auch hinsichtlich der Unterstützung hoher Leistungsanforderungen zu bevorzugen, insbesondere bei lokaler Rechneranordnung (schnelles Kommunikationsmedium). Weiterhin können dabei Rechnerausfälle durch geeignete Recovery-Maßnahmen der DBVS am ehesten für den Benutzer transparent gehalten werden. Föderative DBS liegen bei der Bewertung zwischen den verteilten DC-Systemen und den integrierten Mehrrechner-DBS.

3. Vertikal verteilte Transaktionssysteme

Vertikal verteilte Transaktionssysteme sind durch eine Partitionierung der Systemfunktionen auf Front-End- und Back-End-Rechner gekennzeichnet, welche gemäß einer Client-/Server-Beziehung kooperieren. Bekannteste Vertreter dieser Systemklasse sind Workstation-/Server-Systeme, bei denen gewisse Funktionen zur Transaktionsver-

	Verteilung (nur) im DC-System		*Verteilung im DBS*		
	Programmebene	DML-Ebene	föderative DBS	SN	SD
DBS-Komplexität	+	+	-	-	-
Ortstransparenz	-/o	-	o	++	++
Replikationstransparenz	--	--	o	+	n.a.
Einfachheit d. Programmierung	-	-	o/+	++	++
Flexibilität bezüglich					
Datenallokation	--	-	-	o/+	n.a.
Programmzuordnung	--	o	-	o/+	+
Lastbalancierung	--	--	-	o	+
geographische Verteilung	++	+	+	o	-
Knotenautonomie	++	o	+	-	-
DBS-Heterogenität	++	+	+	-	-
Verfügbarkeit der Daten	-	-	-	+	+
Nutzung von Intra-TA-Parallelität	-	-/o	o	+	+

Bild 2: Qualitative Bewertung horizontal verteilter Transaktionssysteme

arbeitung auf Workstations vorgelagert werden, sowie DB-Maschinen, bei denen eine Auslagerung der DB-Verarbeitung auf Back-End-Rechner erfolgt. In beiden Fällen soll eine Entlastung der allgemeinen Verarbeitungsrechner sowie eine Steigerung der Kosteneffektivität durch verstärkten Einsatz preiswerter (jedoch leistungsfähiger) Mikroprozessoren erfolgen. Für die genannten Systemarchitekturen sprechen daneben weitere Gründe, auf die hier jedoch nicht näher eingegangen werden soll.

Analog zu den horizontal verteilten Transaktionssystemen ergeben sich bei der Partitionierung der Funktionen mehrere Verteilmöglichkeiten im DC-System und DBS, welche unterschiedliche Verteilgranulate mit sich bringen. Im einfachsten Fall liegt eine Kooperation der Front-End (FE)-Rechner mit nur einem Back-End (BE)-Rechner oder Server vor; bei mehreren Server-Rechnern können diese entweder unabhängig voneinander arbeiten oder kooperieren. In letzterem Fall würde das Server-System wiederum ein horizontal verteiltes Transaktionssystem repräsentieren, was die Orthogonalität der beiden Klassifikationsmerkmale verdeutlicht. So ist z.B. bei DB-Maschinen das Back-End-System in der Regel ein integriertes Mehrrechner-DBS (derzeit meist vom Typ SN).

Verteilung im DC-System

Bei der Partitionierung im DC-System werden gewisse Funktionen des TP-Monitors bzw. die Anwendungsprogramme auf Front-End-Rechner vorgelagert. Generell ist mit einer Verkomplizierung der Systemadministration zu rechnen, da nun eine Zuordnung von Systemfunktionen und TAP auf eine potentiell sehr große Anzahl von Front-End-Stationen zu verwalten ist (replizierte Speicherung von Maskendefinitionen, TP-Monitor-Funktionen und TAP, so daß Änderungen entsprechend propagiert werden müssen). Ähnlich wie bei den horizontal verteilten DC-Systemen können drei Partitionierungsmöglichkeiten im DC-System unterschieden werden [6]:

(1) *Vorlagerung der Nachrichtenaufbereitung*
Eine erhebliche Entlastung der Server ergibt sich bereits, wenn Benutzereingaben direkt 'vor Ort' analysiert werden (durch intelligente Terminals, PCs oder Workstations) und in ein neutrales, geräteunabhängiges Nachrichtenformat überführt werden, das von den TAP auf den Servern verarbeitet werden kann. Umgekehrt sollen Ausgabenachrichten beim Benutzer in das gerätespezifische Format überführt werden. Einfache Eingabefehler (z.B. Verletzung von in der Maskendefinition spezifizierten Wertebereichsbeschränkungen) können sofort abgewiesen werden, ohne daß eine Übertragung an den Server erfolgen muß.
Die Ausführung der TAP erfolgt bei diesem Ansatz weiterhin auf dem Server, d.h. die Einheit der Verteilung

ist der gesamte Transaktionsauftrag (wie beim Transaction Routing). Erfolgt auf den Servern keine weitere (horizontale oder vertikale) Verteilung, so ist die Transaktionsausführung auf Daten eines Rechners beschränkt.

(2) *Aufruf von Teilprogrammen auf den Servern (Remote Procedure Call)*
Bei dieser Partitionierungsform werden TAP zum Teil auf den FE-Rechnern (z.B. PCs, Workstations) ausgeführt, so daß diese weitergehend als unter (1) genutzt werden können. Allerdings können die FE-TAP nicht unmittelbar mit den Server-DBVS kommunizieren, sondern müssen Teilprogramme auf den Servern starten, um die dort erreichbaren Daten zu referenzieren. Eine TP-Monitor-Komponente auf den FE-Rechnern ist für die Abwicklung der 'Remote Procedure Calls' zuständig, wobei prinzipiell Funktionen mehrerer unabhängiger und heterogener Server innerhalb eines FE-TAP und damit innerhalb einer Transaktion aufgerufen werden können. In diesem Fall muß ein verteiltes Commit-Protokoll zwischen dem TP-Monitor auf dem FE-Rechner und den TP-Monitoren der involvierten Server-Rechner erfolgen. Da PCs oder Workstations als FE-Rechner als 'unzuverlässig' einzustufen sind, sollte jedoch die Rolle des Commit-Koordinators vom FE auf einen der Server-Rechner übertragen werden.
Die Bewertung des Ansatzes entspricht im wesentlichen der der verteilten Programmierung bei horizontaler Verteilung. Von Nachteil ist die relativ geringe Flexibilität für die TAP-Erstellung auf FE-Seite, da nur über Server-TAP auf die Datenbanken zugegriffen werden kann. Diese Zugriffsform wird u.a. von Sybase und Camelot unterstützt.

(3) *Ausführung von DML-Befehlen auf den Servern*
Hierbei werden die TAP vollständig auf den FE-Rechnern ausgeführt und DML-Befehle zur Ausführung an die Server verschickt. Gegenüber (2) ergeben sich somit größere Freiheitsgrade für die DB-Zugriffe, jedoch auch ein i.a. weit höherer Kommunikationsaufwand, da jede Operation einzeln verschickt werden muß. Weiterhin muß die Schemainformation der Server-Datenbanken auf den FE-Rechnern bekannt sein, und es ist eine aufwendige Definition von Zugriffsrechten erforderlich (an Schemaobjekte und DB-Operationen gebunden). Soll auf Daten mehrerer (unabhängiger) Server zugegriffen werden, sieht der Programmierer mehrere Schemata. Wie bei (2) ist in diesem Fall ein verteiltes Commit-Protokoll durch die TP-Monitore zu unterstützen.
Die einfachste Realisierungsform (Kooperation mit einem Server) wird von vielen relationalen DBVS unterstützt (Oracle, Ingres, Sybase, ...). Einige Systeme (z.B. Ingres) erlauben bereits den Zugriff auf mehrere, heterogene Server/DBVS, wobei derzeit Unterschiede in den Anfragesprachen, Datenmodellen, Zeichendarstellungen, etc. durch Gateways überbrückt werden. Ein Teil des enormen Konvertierungsaufwandes soll durch standardisierte Protokolle wie RDA (Remote Database Access) [8] eliminiert werden.
DB-Maschinen fallen auch in diese Verteilkategorie, wobei hier die TAP i.a. auf allgemeinen Verarbeitungsrechnern ('Hosts') oder auf Workstations ablaufen. Innerhalb des Back-End-Systems sollen v.a. komplexere DB-Operationen durch Parallelverarbeitung effizient abgearbeitet werden. Beispiele von DB-Maschinen sind Teradatas DBC/1024 sowie Prototypen wie Gamma oder EDS.

Verteilung im DBS (vertikal verteilte Mehrrechner-DBS)

Eine weitere Verteilungsmöglichkeit besteht darin, die Anwendungen vollkommen auf den FE-Rechnern auszuführen und die DB-Verarbeitung zwischen FE- und BE-Rechnern zu partitionieren. Diese Realisierungsform wird derzeit v.a. von sogenannten Non-Standard-DBS bzw. objektorientierten DBS verfolgt, wobei durch eine möglichst weitgehende Auslagerung der DB-Verarbeitung auf Workstations ein gutes Antwortzeitverhalten für die komplexen Anwendungsvorgänge in diesen Bereichen (z.B. CAD) angestrebt wird. Insbesondere sollen DB-Objekte nach Bereitstellung durch die Server möglichst im Hauptspeicher der Workstation gepuffert werden (in einem vom Workstation-DBVS verwalteten Objektpuffer), um durch Ausnutzung von Lokalität eine hohe Zugriffsgeschwindigkeit zu erzielen. Wie in [3, 4] näher ausgeführt, kommen für die Kooperation zwischen Workstation- und Server-DBVS wiederum verschiedene Ebenen mit unterschiedlichen Verteileinheiten in Betracht.

Vertikale Verteilung über mehr als zwei Ebenen

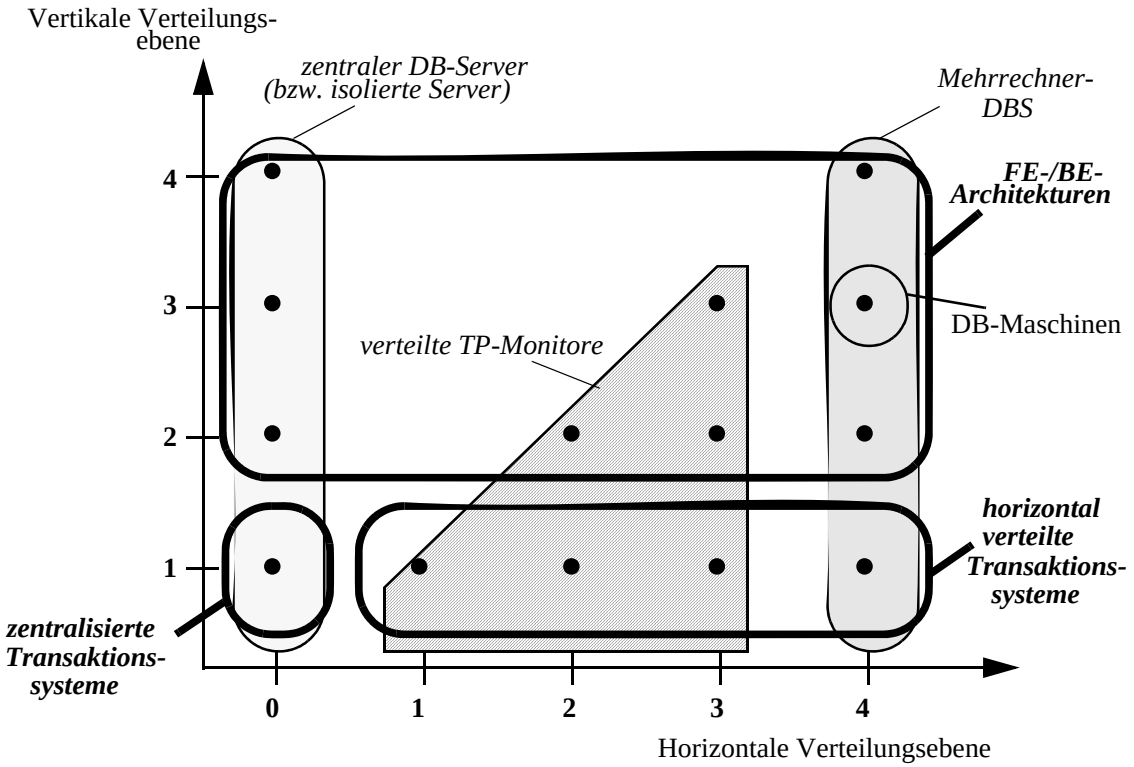
Eine vertikale Verteilung von Systemfunktionen ist nicht auf zwei Ebenen beschränkt, sondern kann über mehrere Stufen erfolgen. Z.B. ist die DC-Verarbeitung auf drei Ebenen aufgeteilt, wenn eine lokale Nachrichtenaufbereitung auf PCs oder Workstations erfolgt, eine Verteilung der Transaktionsaufträge (Transaction Routing) auf Kommunikationsrechnern durchgeführt wird und die TAP-Abarbeitung auf allgemeinen Verarbeitungsrechnern erfolgt; die DB-Verarbeitung könnte dann auf einer weiteren Server-Ebene vorgenommen werden, z.B. im Rahmen einer DB-Maschine. Die DB-Verarbeitung könnte auch über mehrere Stufen partitioniert werden (z.B. zwischen Workstation-DBVS, abteilungsspezifischen DB-Servern und einem zentralen DB-Server eines Unternehmens).

Vergleich

Für die Transaktionsverarbeitung in konventionellen Anwendungen kommen v.a. die Verteilformen 2 und 3 im DC-System in Betracht, für die sich eine weitgehend ähnliche Bewertung wie bei den horizontal verteilten DC-Systemen ergibt (Bild 2). Im allgemeinen ist es wünschenswert, beide Zugriffsformen zu unterstützen, was auch von einigen existierenden Systemen getan wird (z.B. Sybase).

Kombination von horizontaler und vertikaler Verteilung

Wie schon erwähnt, können horizontale und vertikale Verteilformen miteinander kombiniert werden. Das sich dabei ergebende Spektrum verteilter Transaktionssysteme ist in Bild 3 verdeutlicht. Dabei werden sowohl bei horizontaler als auch vertikaler Verteilung die drei eingeführten Verteilebenen im DC-System unterschieden (Verteilung ganzer Transaktionsaufträge, von Teilprogrammen oder DML-Befehlen) sowie als vierte Möglichkeit die Ver-



0 = keine Kooperation
1 = Verteilung auf Transaktionsebene
2 = Aufruf entfernter Teilprogramme (remote procedure call)
3 = Verschicken von DB-Operationen
4 = Kooperation zwischen DBVS
(Verschicken von Teiloperationen, Satz-/Seitenanforderungen, Sperranforderungen, ...)

Bild 3: Kombination horizontal und vertikal verteilter Transaktionssysteme

teilung im DBVS (Mehrrechner-DBS). Bei horizontaler Verteilung wurde eine zusätzliche Verteilebene 0 vorgesehen, die den Fall abdeckt, wenn nur ein einziger Verarbeitungsrechner (Server) bzw. mehrere unabhängige Server vorliegen. (Analog könnte man eine vertikale Verteilebene 0 einführen, welche impliziert, daß die Transaktionen vollständig auf dem FE-Rechner abgearbeitet werden). Zentralisierte Transaktionssysteme sind daher durch die Verteilkombination (0,1) gekennzeichnet, also vertikale Verteilung von ganzen Transaktionsaufträgen (Ebene 1) und keine horizontale Verteilung. Die in Kap. 2 diskutierten Formen horizontal verteilter Systeme entsprechen den Kombinationen (1,1) bis (4,1), während die Kombinationen (0, 1) bis (0,4) den (nur) vertikal verteilten Systemen entsprechen. Die verbleibenden sechs Kombinationen repräsentieren die Systemarchitekturen, bei denen sowohl eine vertikale als eine horizontale Verteilung vorliegt. Dabei ist zu berücksichtigen, daß das horizontale Verteilgranulat stets kleiner oder gleich der vertikalen Verteileinheit sein muß. Wenn also etwa von den FE-Rechnern

DML-Befehle auf den Servern aufgerufen werden, so können dort keine gröberen Granulate wie Teilprogramme oder ganze Transaktionsaufträge mehr verteilt werden. DB-Maschinen werden durch die Kombination (4, 3) repräsentiert, also vertikale Übergabe von DB-Operationen und horizontale Aufteilung innerhalb eines Mehrrechner-DBS. Die komplexeste Realisierung eines verteilten Transaktionssystems liegt vor, wenn sowohl eine vertikale als eine horizontale Kooperation zwischen DBVS durchgeführt wird (Kombination (4,4)).
Existierende Systeme bieten oft mehrere Formen einer verteilten Transaktionsverarbeitung an. Sybase unterstützt z.B. die Kombinationen (0, 2), (0, 3) und (2, 2). Dies bedeutet, man kann von Workstations aus entweder Teilprogramme oder DML-Befehle auf den Servern starten; zusätzlich können innerhalb eines Server-Teilprogrammes Teilprogramme anderer Server aufgerufen werden.

4. Zusammenfassung

Es wurde ein allgemeines Klassifikationsschema für verteilte Transaktionssysteme vorgestellt, das mächtiger als bisherige Klassifikationsansätze [5, 6, 7, 9, 10] ist und einen weitergehenden Vergleich verschiedener Systemarchitekturen unterstützt. Es wurde unterschieden zwischen horizontaler Replikation und vertikaler Partitionierung von Systemfunktionen sowie verschiedenen Verteilungsebenen innerhalb des DC-Systems und des DBVS. Erstmalig wurden auch verschiedenste Kombinationsmöglichkeiten der Verteilung systematisiert. Bei der Beurteilung der Architekturformen wurde u.a. auf die Implikationen für die Allokation von Transaktionsprogrammen, Daten und Lasteinheiten eingegangen. Das vorgestellte Klassifikationsschema kann bei Bedarf verfeinert werden. Z.B. sind für einzelne Systemklassen wie föderative DBS, DB-Maschinen oder Workstation-/Server-DBVS weitergehende Unterscheidungen möglich, wie sie in der Literatur bereits diskutiert wurden.

Die Betrachtung der verschiedenen Ansätze verdeutlichte, daß keine der Systemarchitekturen alle eingangs genannten Anforderungen gut erfüllen kann. Deshalb gibt es auch keine optimale Verteilform, wie gelegentlich suggeriert wird, sondern es wird weiterhin eine Koexistenz verschiedener Architekturansätze und Kooperationsformen zur verteilten Transaktionsverarbeitung geben. So werden Forderungen nach hoher Knotenautonomie, geographischer Verteilung und Unterstützung heterogener Datenbanken am besten durch verteilte DC-Systeme oder föderative DBS erfüllt (s. auch [2]). Integrierte Mehrrechner-DBS bieten dagegen Vorteile hinsichtlich Ortstransparenz, einfacher Anwendungsprogrammierung und Datenverfügbarkeit. Forderungen nach hoher Leistungsfähigkeit werden am ehesten von lokal gekoppelten Mehrrechner-DBS erfüllt, wo ein schnelles Kommunikationsmedium genutzt werden kann und die besten Möglichkeiten zur Parallelisierung komplexer Anfragen und einer dynamischen Lastbalancierung bestehen. Die Vorlagerung von DC-Funktionen auf PCs oder Workstations wird zunehmend an Bedeutung gewinnen, da sie bessere Benutzeroberflächen und eine verbesserte Kosteneffektivität ermöglicht.

5. Literatur

[1] Special Issue on Heterogeneous Databases. ACM Computing Surveys 22 (3), 175-293 (1990)

[2] Butsch, K., Rahm, E.: Architekturansätze zur Unterstützung heterogener Datenbanken. Univ. Kaiserslautern, FB Informatik (1991)

[3] Deppisch, U. et al.: Überlegungen zur Datenbank-Kooperation zwischen Server und Workstations. Proc. 16. GI-Jahrestagung, Informatik-Fachberichte 126,Springer-Verlag, 565-580 (1986)

[4] DeWitt, D.J., Futtersack, P., Maier, D., Velez, F.: A Study of Three Alternative Workstation-Server Architectures for Object Oriented Database Systems. Proc. 16th VLDB conf., 107-121 (1990)

[5] Effelsberg, W.: Datenbankzugriff in Rechnernetzen. Informationstechnik 29 (3), 140-153 (1987)

[6] Härder, T., Meyer-Wegener, K.: Transaktionssysteme in Workstation/Server-Umgebungen. Informatik Forschung und Entwicklung 5, 127-143 (1990)

[7] Härder, T., Rahm, E.: Mehrrechner-Datenbanksysteme für Transaktionssysteme hoher Leistungsfähigkeit. Informationstechnik 28 (4), 214-225 (1986)

[8] Lamersdorf, W.: Remote Database Access: Kommunikationsunterstützung für Fernzugriff auf Datenbanken. Informatik-Spektrum (Das aktuelle Schlagwort) 14 (3), 161-162, (1991)

[9] Leff, A., Pu, C.: A Classification of Transaction Processing Systems. IEEE Computer, 63-76 (June 1991)

[10] Meyer-Wegener, K.: Transaktionssysteme - verteilte Verarbeitung und verteilte Datenhaltung. Informationstechnik 29 (3), 120-126 (1987)

[11] Meyer-Wegener, K.: Transaktionssysteme. Teubner-Verlag (1988)

[12] Rahm, E.: Der Database-Sharing-Ansatz zur Realisierung von Hochleistungs-Transaktionssystemen, Informatik-Spektrum 12 (2), 65-81 (1989)

[13] Rahm, E.: A Framework for Workload Allocation in Distributed Transaction Processing Systems. ZRI-Bericht 13/89, Univ. Kaiserslautern, erscheint in: The Journal of Systems and Software (1992).

[14] Schek, H.-J., Weikum, G.: Erweiterbarkeit, Kooperation, Föderation von Datenbanksystemen. Proc. BTW-91, Informatik-Fachberichte 270, Springer-Verlag, 38-71 (1991)