(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 699 17 333 T2** 2005.05.12

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 162 538 B1**
(21) Deutsches Aktenzeichen: **699 17 333.7**
(96) Europäisches Aktenzeichen: **01 119 226.7**
(96) Europäischer Anmeldetag: **12.02.1999**
(97) Erstveröffentlichung durch das EPA: **12.12.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **12.05.2004**
(47) Veröffentlichungstag im Patentblatt: **12.05.2005**

(51) Int Cl.[7]: **G06F 11/14**
G06F 9/46, G06F 12/08

(30) Unionspriorität:

74587 P	**13.02.1998**	**US**
199120	**24.11.1998**	**US**

(84) Benannte Vertragsstaaten:
DE, FR, GB, NL

(73) Patentinhaber:
Oracle Corp., Redwood Shores, Calif., US

(74) Vertreter:
Viering, Jentschura & Partner, 80538 München

(72) Erfinder:
Bamford, Roger J., San Francisco, US; Klots, Boris, Belmont, US

(54) Bezeichnung: **Übertragung einer Ressource von einem ersten Zwischenspeicher an einen zweiten Zwischenspeicher**

Beschreibung

Gebiet der Erfindung

[0001] Die Erfindung betrifft Techniken zum Reduzieren der Nachteile, welche damit assoziiert sind, wenn ein Knoten Daten von einem Datenspeicher anfordert, wenn sich die neueste Version der angeforderten Daten im Zwischenspeicher eines anderen Knotens befindet.

Hintergrund der Erfindung

[0002] Um die Skalierbarkeit zu verbessern, erlauben manche Datenbank-Systeme es mehr als einem Datenbankserver (von denen jeder separat läuft) konkurrierend auf gemeinsam benutzten Speicher, wie sie zum Beispiel auf Plattenmedien gespeichert sind, zuzugreifen. Jeder Datenbankserver hat einen Zwischenspeicher zum Zwischenspeichern gemeinsam benutzter Ressourcen, wie zum Beispiel Plattenblöcke. Solche Systeme werden hierin als Parallel-Server-Systeme bezeichnet.

[0003] Ein Problem, welches mit Parallel-Server-Systemen assoziiert ist, ist die Möglichkeit für sogenannte „Pings". Ein Ping tritt auf, wenn die Version einer Ressource, welche sich in dem Zwischenspeicher eines Servers befindet, zu dem Zwischenspeicher eines anderen Servers geliefert werden muss. Dadurch tritt ein Ping auf, sobald, nachdem ein Datenbankserver A eine Ressource x in seinem Zwischenspeicher modifiziert hat, ein Datenbankserver B die Ressource x zum modifizieren anfordert. Datenbankserver A und B werden typischerweise auf verschiedenen Knoten laufen, können aber auch in manchen Fällen auf demselben Knoten laufen.

[0004] Ein Ansatz, Pings zu handhaben wird hierin als der „Platten-Interventions"-Ansatz bezeichnet. Der Platten-Interventions-Ansatz verwendet eine Platte als intermediären Speicher, um die letzte Version der Ressource zwischen zwei Zwischenspeichern zu übertragen. Daher erfordert der Platten-Interventions-Ansatz, im oben beschriebenen Beispiel, Datenbankserver 1, um die Version der Ressource X aus seinem Zwischenspeicher auf Platte zu schreiben und Datenbankserver 2, um diese Version von Platte in seinen Zwischenspeicher zu laden. Das Stützen des Platten-Interventions-Ansatzes auf zwei Platten E/As je Zwischenserver-Übertragung einer Ressource limitiert die Skalierbarkeit von Parallel-Server-Systemen. Insbesondere sind die Platten E/As, welche benötigt werden, um ein Ping zu handhaben, relativ teuer und zeitraubend und je mehr Datenbankserver zu dem System hinzugefügt werden, desto höher wird die Anzahl der Pings.

[0005] Jedoch sorgt der Platten-Interventions-Ansatz für eine relativ effiziente Wiederherstellung von Einzel-Datenbankserver-Ausfällen, indem solches Wiederherstellen nur das Ausführen des Wiederherstellungsprotokolls (Wiederholungsprotokoll) des abgestürzten Datenbankservers benötigt. Das Ausführen des Wiederholungsprotokolls des abgestürzten Datenbankservers stellt sicher, dass alle durchgeführten Änderungen, welche Transaktionen auf dem abgestürzten Datenbankserver an den Ressourcen in dem Zwischenspeicher des abgestürzten Server gemacht haben, wiederhergestellt werden. Das Verwenden von Wiederholungsprotokollen während einer Wiederherstellung ist im Detail in der U.S. Patentanmeldung Nr. 08/784,611 beschrieben mit dem Titel „Caching Data in Recoverable Objects", eingereicht am 21. Januar 1997.

[0006] Parallel-Server-Systeme, welche den Platten-Interventions-Ansatz anwenden, verwenden typischerweise ein Protokoll, in welchem jede globale Entscheidung, welche Ressourcenzugriff und Ressourcenmodifikationen betrifft, mittels eines Verteilten-Sperrmanagers (DLM) durchgeführt wird. Die Funktion eines beispielhaften DLM ist im Detail in der am 24. Juni 1996 eingereichten U.S. Patentanmeldung Nummer 08/669,689 mit dem Titel „Method and Apparatus for Lock Caching" beschrieben.

[0007] In typischen Verteilten-Sperrmanager-Systemen werden Informationen, welche irgendeine gegebene Ressource betreffen, in einem Sperr-Objekt gespeichert, welches mit der Ressource korrespondiert. Jedes Sperr-Objekt ist in dem Speicher eines einzelnen Knoten gespeichert. Der Sperrmanager, welcher in dem Knoten, in welchem ein Sperr-Objekt gespeichert ist, residiert, wird als Hauptserver dieses Sperr-Objekts und der Ressource, die es umfasst, bezeichnet.

[0008] In Systemen, welche den Platten-Interventions-Ansatz zum Handhaben von Pings anwenden, tendieren Pings dazu, den DLM in eine Vielzahl von sich auf Sperren beziehende Kommunikationen zu verwickeln. Insbesondere, wenn ein Datenbankserver (der „anfordernde Server") auf eine Ressource zugreifen muss, prüft der Datenbankserver, ob er die gewünschte Ressource in dem passenden Modus gesperrt hat: entweder gemeinsam im Falle eines Lesens, oder ausschließlich im Falle eines Schreibens. Wenn der anfordernde Datenbankserver die gewünschte Ressource nicht in dem richtigen Modus gesperrt hat oder keinerlei Sperre auf die Ressource hat, dann sendet der anfordernde Server eine Anforderung zu dem Hauptserver für die Ressource, um die Sperre im spezifiziertem Modus zu erlangen.

[0009] Die Anforderung, welche von dem anfordernden Datenbankserver veranlasst wird, kann mit dem aktuellen Status der Ressource im Konflikt stehen (z. B. könnte es einen anderen Datenbankserver geben, welcher jetzt eine Ausschließlich-Sperre auf die Res-

source hält). Wenn es keinen Konflikt gibt, erteilt der Hauptserver der Ressource die Sperre und speichert die Erteilung. Im Falle eines Konflikts initiiert der Hauptserver der Ressource ein Konflikt-Lösungs-Protokoll. Der Hauptserver der Ressource instruiert den Datenbankserver, welcher die im Widerspruch stehende Sperre hält (den „Halter") seine Sperre in einen niedrigeren, kompatiblen Modus herabzusetzen.

[0010] Unglücklicherweise kann, wenn der Halter (z. B. der Datenbankserver A) jetzt eine aktualisierte („unreine") Version der gewünschten Ressource in seinem Zwischenspeicher hat, er nicht unmittelbar seine Sperre herabsetzen. Um seine Sperre herabzusetzen, durchläuft der Datenbankserver A etwas, was als „Hard-Ping" Protokoll bezeichnet wird. Gemäß dem Hard-Ping Protokoll, erzwingt Datenbankserver A, dass das Wiederholungsprotokoll, welches mit dem Aktualisieren assoziiert ist, auf die Platte geschrieben wird, schreibt die Ressource auf Platte, setzt seine Sperre herab und meldet dem Hauptserver, dass Datenbankserver A fertig ist. Nach Empfangen der Benachrichtigung registriert der Hauptserver das Erteilen der Sperre und benachrichtigt den anfordernden Server, dass die angeforderte Sperre erteilt wurde. Zu diesem Punkt liest der anfordernde Server B die Ressource von der Platte in seinen Zwischenspeicher.

[0011] Wie oben beschrieben erlaubt der Platten-Interventions-Ansatz nicht, dass eine Ressource, welche mittels eines Datenbankserver aktualisiert wurde (eine „unreine Ressource"), direkt an einen anderen Datenbankserver versendet wird. Solch direktes Versenden wird wegen Problemen, welche mit dem Wiederherstellen verbunden sind, als undurchführbar betrachtet. Beispielsweise wird angenommen, dass eine Ressource am Datenbankserver A modifiziert wird und dann direkt zum Datenbankserver B versendet wird. Am Datenbankserver B wird die Ressource ebenfalls modifiziert und dann zum Datenbankserver A zurückgesendet. Im Datenbankserver A wird die Ressource ein drittes Mal modifiziert. Ferner wird angenommen, dass jeder Server alle Wiederholungsprotokolle auf Platte schreibt, bevor er die Ressource zu einem anderen Server sendet, um es dem Empfänger zu ermöglichen, sich auf frühere Änderungen zu verlassen.

[0012] Nach der dritten Aktualisierung wird angenommen, dass der Datenbankserver A abstürzt. Das Protokoll des Datenbankservers A enthält Datensätze von Modifikationen an der Ressource mit einer Lücke. Insbesondere enthält das Protokoll des Servers A nicht solche Modifikationen, welche vom Datenbankserver B gemacht wurden. Vielmehr sind die Modifikationen, welche vom Server B gemacht wurden, in dem Protokoll des Datenbankservers B gespeichert. An diesem Punkt müssen, um die Ressource wiederherzustellen, die zwei Protokolle erst zusammengefügt werden, bevor sie angewandt werden. Diese Protokoll-Zusammenfüg-Operation würde, wenn realisiert, Zeit und Ressourcen proportional zu der Gesamtenzahl von Datenbankservern, inklusive derer die nicht abgestürzt sind, benötigen.

[0013] Der oben beschriebene Platten-Interventions-Ansatz vermeidet das Problem, welches mit dem Zusammenfügen von Wiederherstellungsprotokollen nach einem Absturz verbunden ist, aber verschlechtert die Leistungsfähigkeit von sich im Dauerzustand (Normalzustand) befindenden Parallel-Server-Systemen zu Gunsten einer leichten und effizienten Wiederherstellung. Der Direkt-Versendungs-Ansatz vermeidet den Overhead, welcher mit dem Platten-Interventions-Ansatz assoziiert ist, führt aber im Falle von Abstürzen zu komplexen und nicht skalierbaren Wiederherstellungs-Operationen.

[0014] Das Dokument US-A-5,327,556 lehrt eine schnelle Technik zum Übertagen von Dateneinheiten zwischen Übertragungssystemen innerhalb einer gemeinsam genutzten Plattenumgebung, wobei unreine Seiten von einem besitzenden System zu einem anfordernden System übertragen werden, ohne es auf Platte zu schreiben.

[0015] Das Dokument AHMED R. E. ET AL.: "Cache-Aided Roll-Back Error Recovery (CARER) Algorithms for Shared-Memory Multiprocessor Systems", INTERNATIONAL SYMPOSIUM ON FAULT TOLERANT COMPUTING (FTCS), LOS ALAMITOS, US, 26.–28. Juni 1990, IEEE COMP. SOC. PRESS, Band 20, Seiten 82–88, 26. Juni 1990, lehrt verschiedene zwischenspeicherunterstützte Roll-Back-Wiederherstellungs-Techniken zum Gebrauch in Shared-Memory-Multiprozessorsystemen.

[0016] Basierend auf dem Vorhergehenden ist es anschaulich wünschenswert ein System und Verfahren zum Reduzieren des Overheads, welcher mit einem Ping assoziiert ist, bereitzustellen, ohne die Komplexität und die Dauer der Wiederherstellungsoperationen stark zu erhöhen.

Zusammenfassung der Erfindung

[0017] Gemäß der Erfindung, welche im Detail in den anhängenden unabhängigen Ansprüchen 1, 22 und 23 definiert ist, werden ein Verfahren, eine Vorrichtung sowie ein computerlesbares Medium, welches Computer ausführbare Anweisungen trägt, zum Übertragen einer Ressource von dem Zwischenspeicher eines Datenbankservers zu dem Zwischenspeicher eines anderen Datenbankservers, ohne die Ressource zuerst auf Platte zu schreiben, bereitgestellt. Wenn ein Datenbankserver (Anforderer) wünscht, eine Ressource zu modifizieren, fragt der Anforderer nach der aktuellen Version der Ressour-

ce. Der Datenbankserver, welcher die aktuelle Version hat (Halter), versendet die aktuelle Version direkt an den Anforderer. Nach Versenden der Version, verliert der Halter die Genehmigung, die Ressource zu modifizieren, aber behält weiterhin eine Kopie der Ressource im Speicher. Wenn die behaltene Version der Ressource oder eine spätere Version davon auf Platte geschrieben wird, kann der Halter die behaltene Version der Ressource verwerfen. Anderenfalls verwirft der Halter die behaltene Version nicht. Im Falle eines Serverabsturzes, werden, so notwendig, die früheren Kopien aller Ressourcen mit Modifikationen in dem Wiederherstellungsprotokoll des abgestürzten Servers als Startpunkt zum Ausführen des Wiederherstellungsprotokolls des abgestürzten Servers verwendet. Unter Verwenden dieser Technik werden Einzel-Server-Abstürze (die üblichste Form von Abstürzen) wiederhergestellt, ohne die Wiederherstellungsprotokolle der verschiedenen Datenbankserver, welche Zugriff auf die Ressource hatten, zusammenfügen zu müssen.

Kurzbeschreibung der Figuren

[0018] Die Erfindung ist mittels Beispiels und nicht mittels Einschränkung anhand der Figuren der beiliegenden Zeichnungen erläutert, in welchen gleiche Bezugszeichen sich auf ähnliche Elemente beziehen und in welchen:

[0019] Fig. 1 ein Blockdiagramm ist, welches Zwischenspeicher zu Zwischenspeicher Übertragungen der neuesten Versionen von Ressourcen erläutert;

[0020] Fig. 2 ein Flussdiagramm ist, welches Schritte zum Übertragen einer Ressource von einem Zwischenspeicher zu einem anderen, ohne Platten-Intervention gemäß einem Ausführungsbeispiel der Erfindung erläutert;

[0021] Fig. 3 ein Flussdiagramm ist, welches Schritte zum Freigeben früherer Abbilder von Ressourcen gemäß einem Ausführungsbeispiel der Erfindung erläutert;

[0022] Fig. 4 ein Flussdiagramm ist, welches Schritte zum Wiederherstellen nach einem Absturz eines einzelnen Servers gemäß einem Ausführungsbeispiel der Erfindung erläutert;

[0023] Fig. 5 ein Blockdiagramm ist, welches einen Prüfpunkt-Zyklus, gemäß einem Ausführungsbeispiel der Erfindung, erläutert;

[0024] Fig. 6 ein Blockdiagramm eines Computersystems ist, auf welchem ein Ausführungsbeispiel der Erfindung realisiert werden kann.

Detaillierte Beschreibung des bevorzugten Ausführungsbeispiels

[0025] Es werden ein Verfahren und eine Vorrichtung zum Reduzieren des Overheads, welcher mit einem Ping assoziiert ist, beschrieben. In der folgenden Beschreibung werden zum Zweck der Erläuterung zahlreiche spezifische Details erklärt, um ein vollständiges Verständnis dieser Erfindung zu schaffen. Es wird jedoch für einen Fachmann ersichtlich sein, dass diese Erfindung ohne diese spezifischen Details umgesetzt werden kann. In anderen Datenbankservern werden wohl bekannte Strukturen und Vorrichtungen in Blockdiagrammform gezeigt, um unnötiges Verschleiern dieser Erfindung zu verhindern.

Funktioneller Überblick

[0026] Gemäß einem Aspekt der Erfindung werden Pings mittels direkten Versendens von aktualisierten Versionen der Ressourcen, ohne zuerst auf Platte gespeichert zu werden, direkt zwischen Datenbankservern gehandhabt, wodurch der E/A Overhead, welcher mit dem Platten-Interventions-Ansatz assoziiert ist, vermieden wird. Ferner werden die Schwierigkeiten, welche mit dem Wiederherstellen eines Einzelinstanz-Absturzes assoziiert sind, vermieden, indem, sogar wenn die Ressource zu einem anderen Zwischenspeicher übertragen wurde, verhindert wird, dass eine modifizierte Version der Ressource im Zwischenspeicher ersetzt wird, bis die modifizierte Ressource oder irgendein Nachfolger davon auf Platte geschrieben wurde.

[0027] Zum Zweck der Erläuterung wird eine Kopie einer Ressource, welche im Zwischenspeicher nicht ersetzt werden kann, hierin als eine „festgehaltene" Ressource bezeichnet. Die Handlung, eine festgehaltene Ressource ersetzbar zu machen, wird als „Freigeben" der Ressource bezeichnet.

Der M- und W-Sperren-Ansatz

[0028] Gemäß einem Aspekt der Erfindung sind die Genehmigungen zum Modifizieren und Auf-Platte-Schreiben für eine Ressource getrennt. Daher hat ein Datenbankserver, welcher die Genehmigung hat, eine aktualisierte Version einer Ressource vom Zwischenspeicher auf Platte zu schreiben, nicht notwendigerweise die Genehmigung, die Ressource zu aktualisieren. Umgekehrt hat ein Datenbankserver, welcher die Genehmigung hat, eine zwischengespeicherte Version einer Ressource zu modifizieren, nicht Notwendigerweise die Genehmigung, diese zwischengespeicherte Version auf Platte zu schreiben.

[0029] Gemäß einem Ausführungsbeispiel wird diese Trennung der Genehmigungen mittels des Verwendens von speziellen Sperren durchgesetzt. Insbesondere kann die Genehmigung, eine Ressource

zu modifizieren, mittels einer „M"-Sperre erteilt werden, während die Genehmigung, eine Ressource auf Platte zu schreiben, mittels einer „W"-Sperre erteilt werden kann. Es sollte jedoch angemerkt werden, dass das Benutzen von M- und W-Sperren wie hierin beschrieben aber nur einen Mechanismus darstellt, welcher eine übertragene Version einer Ressource davor schützt, im Zwischenspeicher ersetzt zu werden, bis diese Version oder ein Nachfolger davon auf Platte geschrieben wurde.

[0030] Bezugnehmend auf Fig. 2 erläutert sie die Schritte, welche in Antwort auf ein Ping in einem Datenbank-System, welches M- und W-Sperren gemäß einem Ausführungsbeispiel der Erfindung, verwendet, ausgeführt werden. In Schritt **200** fordert ein Datenbankserver, welcher wünscht, eine Ressource zu modifizieren, die M-Sperre vom Hauptserver der Ressource (d. h. dem Datenbankserver, welcher die Sperren der Ressource verwaltet) an. In Schritt **202** instruiert der Hauptserver den Datenbankserver, welcher gerade die M-Sperre der Ressource hält („der Halter"), die M-Sperre zusammen mit der zwischengespeicherten Version der Ressource mittels direkter Übertragung über den Kommunikationskanal (-kanäle), welche(r) die zwei Server koppelt(n) (die „Verbindung"), zu dem anfordernden Datenbankserver zu übertragen.

[0031] In Schritt **204** schickt der Halter die aktuelle Version der Ressource und die M-Sperre zu dem Anforderer. In Schritt **206**, informiert der Halter den Hauptserver über die Übertragung der M-Sperre. In Schritt **208** aktualisiert der Hauptserver die Sperren-Information für die Ressource, um anzuzeigen, dass der Anforderer nun die M-Sperre hält.

PI-Ressourcen

[0032] Der Halter der M-Sperre hat nicht notwendigerweise die W-Sperre und mag daher nicht die Genehmigung haben, die Version der Ressource, welche in seinem Zwischenspeicher enthalten ist, auf Platte heraus zu schreiben. Der übertragende Datenbankserver (d. h. der Datenbankserver, welcher zuletzt die M-Sperre hielt) setzt daher fort, seine Version der Ressource im dynamischen Speicher festzuhalten, weil er, wie unten beschrieben, zu einem zukünftigen Zeitpunkt aufgefordert werden kann, seine Version herauszuschreiben. Die Version der Ressource, welche in dem übertragenden Datenbankserver verbleibt, wird veraltet werden, wenn der empfangende Datenbankserver seine Kopie der Ressource modifiziert. Der übertragende Datenbankserver wird nicht notwendigerweise wissen, wann der empfangende Datenbankserver (oder ein Nachfolger davon) die Ressource modifiziert, daher behandelt er seine verbleibende Version von der Zeit an, wo der übertragende Datenbankserver eine Kopie der Ressource schickt, als „möglicherweise veraltet". Solche möglicherweise veralteten Versionen einer Ressource werden hierin als Frühere-Abbild-Ressourcen (PI-Ressourcen) bezeichnet.

Freigeben von PI-Ressourcen

[0033] Nachdem eine zwischengespeicherte Version einer Ressource freigegeben ist, kann sie mit neuen Daten überschrieben werden. Typischerweise kann eine unreine Version einer Ressource mittels Schreibens der Ressource auf Platte freigegeben werden. Jedoch haben Datenbankserver mit PI-Ressourcen im Zwischenspeicher nicht Notwendigerweise das Recht, PI-Ressourcen auf Platte zu speichern. Eine Technik zum Freigeben von PI-Ressourcen unter diesen Umständen ist in Fig. 3 erläutert.

[0034] Bezugnehmend auf Fig. 3, schickt ein Datenbankserver, wenn er wünscht, eine PI-Ressource in seinem Zwischenspeicher freizugeben, eine Anforderung für die W-Sperre (Schritt **300**) an den Verteilten-Sperrmanager (DLM). Im Schritt **302** beauftragt der DLM dann den anfordernden Datenbankserver, oder irgendeinen Datenbankserver, welcher eine neuere Version der Ressource (ein Nachfolger) in seinem Zwischenspeicher hat, die Ressource auf Platte herauszuschreiben. Dem Datenbankserver, welcher so aufgefordert wird, die Ressource auf Platte zu schreiben, wird die W-Sperre erteilt. Nachdem der Datenbankserver, welchem die W-Sperre erteilt wurde, die Ressource auf Platte geschrieben hat, gibt der Datenbankserver die W-Sperre frei.

[0035] Der DLM sendet dann eine Nachricht zu allen Datenbankservern aus, welche die Version der Ressource anzeigt, welche herausgeschrieben wurde (Schritt **304**), so dass alle früheren PI-Versionen der Ressource freigegeben werden können (Schritt **306**). Beispielsweise wird angenommen dass die Version, welche zur Zeit T10 modifiziert wurde, auf Platte geschrieben wurde. Ein Datenbankserver mit einer Version der Ressource, welche zuletzt zu einer früheren Zeit T5 modifiziert wurde, könnte jetzt den Puffer, in welchem sie gespeichert ist, für andere Daten verwenden. Ein Datenbankserver mit einer Version, welche zu einer späteren Zeit T11 modifiziert wurde, müsste jedoch fortfahren, seine Version der Ressource in seinem Speicher zu halten.

Ping Management unter dem M- und W-Sperren-Ansatz

[0036] Gemäß einem Ausführungsbeispiel der Erfindung, kann, um Pings zu handhaben, der M- und W-Sperren-Ansatz, wie er nun mit Bezug auf Fig. 1 beschrieben werden soll, implementiert werden. Bezugnehmend auf Fig. 1 ist es ein Blockschaltbild, welches vier Datenbankserver A, B, C, und D erläutert, von denen alle Zugang zu einer Datenbank haben, welche eine spezielle Ressource enthält. Zu

dem dargestellten Zeitpunkt hat jeder der Datenbankserver A, B und C Versionen der Ressource. Die Version, welche im Zwischenspeicher des Datenbankservers A gehalten wird, ist die vor der kürzesten Zeit modifizierte Version der Ressource (modifiziert zur Zeit T10). Die Versionen, welche in den Datenbankservern B und C gehalten werden, sind PI-Versionen der Ressource. Datenbankserver D ist der Hauptserver der Ressource.

[0037] Zu diesem Punkt wird angenommen dass ein anderer Datenbankserver (der „Anforderer") wünscht, die Ressource zu modifizieren. Der Anforderer fordert die Modifizier-Sperre vom Hauptserver an. Der Hauptserver schickt einen Befehl zum Datenbankserver A, um die Sperre wegen der im Widerspruch stehenden Anforderung des Anforderers herabzukonvertieren (ein „BAST"). In Erwiderung auf den Herabkonvertierungsbefehl, wird das aktuelle Abbild der Ressource (egal ob sauber oder unrein) vom Datenbankserver A zusammen mit der Genehmigung, die Ressource zu modifizieren, zu dem Anforderer gesandt. Die so gesandte Genehmigung schließt keine Genehmigung ein, die Ressource auf Platte zu schreiben.

[0038] Wenn Datenbankserver A die M-Sperre zu dem Anforderer weiterreicht, setzt Datenbankserver A seine M-Sperre zu einer „Halte"-Sperre (und „H-Sperre") herab. Die H-Sperre zeigt an, dass der Datenbankserver A eine festgehaltene PI-Kopie hält. Besitz einer H-Sperre verpflichtet den Besitzer die PI-Kopie in seinem Puffer-Zwischenspeicher zu halten, aber gibt dem Datenbankserver keinerlei Rechte die PI-Kopie auf Platte zu schreiben. Es kann mehrere gleichzeitige H-Halter für die gleiche Ressource geben, aber zu einer Zeit kann nicht mehr als ein Datenbankserver die Ressource schreiben, daher kann nur ein Datenbankserver eine W-Sperre an der Ressource halten.

[0039] Bevor die Ressource versendet wird, stellt der Datenbankserver A sicher, dass das Protokoll durchgesetzt ist (d. h. dass das Wiederherstellungsprotokoll, welches für die Änderungen der Ressource erzeugt wurde, welche mittels des Datenbankservers A gemacht wurden, dauerhaft gespeichert wird). Mittels Weiterreichens der Modifikations-Genehmigung, verliert Datenbankserver A sein eigenes Recht die Ressource zu modifizieren. Die Kopie der Ressource (wie sie gerade zum Zeitpunkt des Versendens war) wird immer noch im verschickenden Datenbankserver A gehalten. Nach dem Verschicken der Ressource ist die Kopie der Ressource, welche im Datenbankserver A gehalten wird, eine PI-Ressource.

Gefälligkeits-Schreiben

[0040] Nachdem ein Datenbankserver eine unreine Ressource direkt zu einem anderen Datenbankserver geschickt hat, wird die zurückgehaltene Kopie der Ressource eine festgehaltene PI-Ressource, deren Puffer, bis sie freigegeben wird, nicht für eine andere Ressource genutzt werden kann. Die Puffer, welche PI-Ressourcen enthalten werden hierin als PI Puffer bezeichnet. Diese Puffer belegen wertvollen Platz in den Zwischenspeichern der Datenbankserver und müssen letztlich für andere Daten wiederverwendet werden.

[0041] Um PI-Puffer im Puffer-Zwischenspeicher (welcher als veraltet gekennzeichnet oder zu überprüfen ist) zu ersetzen, wird ein neues Platten-Schreib-Protokoll angewendet, welches hierin als „Gefälligkeits-Schreiben" bezeichnet wird. Gemäß dem Gefälligkeits-Schreib-Protokoll sendet der Datenbankserver, wenn ein Datenbankserver eine Ressource auf Platte schreiben muss, die Anforderung an den DLM. Der DLM wählt eine Version der Ressource aus, welche auf Platte geschrieben werden soll, findet den Datenbankserver, welcher die ausgewählte Version hat und veranlasst, im Interesse desjenigen Datenbankservers, welcher die Schreibanforderung initiiert hat, diesen Datenbankserver, die Ressource auf Platte zu schreiben. Der Datenbankserver, welcher die Ressource wirklich auf Platte schreibt, kann, abhängig von der letzten Trajektorie der Ressource, der Datenbankserver sein, welcher das Schreiben angefordert hat, oder irgendein anderer Datenbankserver.

[0042] Schreiben der ausgewählten Version einer Ressource auf Platte gibt alle PI-Versionen, welche gleich alt oder älter als die ausgewählte Version sind, der Ressource, welche auf Platte geschrieben wurde, in allen Puffer-Zwischenspeichern eines Clusters frei. Die Kriterien, welche verwendet werden, um die Version zu wählen, welche auf Platte geschrieben wird, sollen nachfolgend in größerem Detail beschrieben werden. Jedoch kann die ausgewählte Version entweder die neueste PI-Version, welche dem Hauptserver bekannt ist, oder die aktuelle Version („CURR") der Ressource sein. Ein Vorteil des Auswählens einer anderen Version als der aktuellen Version ist, dass die Auswahl einer anderen Version die aktuelle Kopie ununterbrochen für Modifikationen verfügbar lässt.

[0043] Ein Datenbankserver, welcher eine PI-Ressource hält, kann, vorrausgesetzt, dass er eine W-Sperre auf die Ressource erlangt hat, seine PI-Kopie herausschreiben. Die Schreibvorgänge der Ressource sind von der Migration des CURR-Ressourcen-Abbildes zwischen den verschiedenen Datenbankservern entkoppelt.

Effektivitätsfaktoren

[0044] Es besteht keine Notwendigkeit, jedes mal, wenn eine Ressource zu einem anderen Datenbank-

server verschickt wird, eine PI-Kopie zu schreiben. Folglich ist das Ziel vom dauerhaften Speichern von Ressourcen, die Plattenkopien aktuell genug zu halten und die Anzahl von nicht-überschreibbaren Ressourcen in den Puffer-Zwischenspeichern vernünftig zu halten. Verschiedene Faktoren bestimmen die Effizienz eines Systems, welches das oben beschriebene Gefälligkeits-Schreib-Protokoll anwendet. Insbesondere ist es wünschenswert:

(1) E/A Aktivität, welche durch das Schreiben von unreinen Ressourcen auf Platte verursacht wird, zu minimieren

(2) die Plattenversionen der Ressourcen ausreichend aktuell halten, um die Wiederherstellungs-Operationen nach einem Absturz zu beschleunigen; und

(3) Überlauf des Puffer-Zwischenspeichers mit festgehaltenen PI-Ressourcen zu verhindern.

[0045] Maximieren des ersten Kriteriums hat einen negativen Einfluss auf das zweite und dritte Kriterium und umgekehrt. Dadurch ist ein Kompromiss notwendig. Gemäß einem Ausführungsbeispiel der Erfindung kann, gekoppelt mit einer Kontrolle über den gesamten E/A-Haushalt, ein selbst-abstimmender Algorithmus, welcher verschiedene Techniken von mit Prüfpunkten versehen kombiniert (LRU gemischt mit zufälligen kontinuierlichen mit Prüfpunkten versehen), verwendet werden.

Der Aktuell-Schreiben-Ansatz

[0046] Eine Alternative zu dem oben beschriebenen Gefälligkeits-Schreib-Protokoll wird hierin als der Aktuell-Schreiben-Ansatz bezeichnet. Gemäß dem Aktuell-Schreiben-Ansatz haben alle Datenbankserver die Genehmigung ihre PI-Ressourcen auf Platte zu schreiben. Bevor sie dies jedoch machen, erlangt ein Datenbankserver eine Sperre auf die plattenresidente Kopie der Ressource. Nach Erlangen der Sperre vergleicht der Datenbankserver die Plattenversion mit der PI-Version, welche er zu schreiben wünscht. Wenn die Plattenversion älter ist, dann wird die PI-Version auf Platte geschrieben. Wenn die Plattenversion neuer ist, dann kann die PI-Version abgelegt werden und der Puffer, welcher belegt ist, kann wiederverwendet werden.

[0047] Im Gegensatz zu dem Gefälligkeits-Schreib-Protokoll erlaubt der Aktuell-Schreiben-Ansatz einem Datenbankserver seine eigene PI-Version freizugeben, entweder indem er sie auf Platte schreibt oder indem er feststellt, dass die Plattenversion neuer ist. Jedoch erhöht der Aktuell-Schreiben-Ansatz den Wettstreit um die Sperre der plattenresidenten Kopie und kann ein Platten-E/A nach sich ziehen, welcher mit dem Gefälligkeits-Schreib-Ansatz nicht aufgetreten wäre.

Genehmigungsketten

[0048] Typische DLMs verwalten den Zugang zu Ressourcen mittels des Verwendens einer begrenzten Anzahl von Sperrmodi, wobei die Modi entweder kompatibel sind oder im Konflikt stehen. Gemäß einem Ausführungsbeispiel ist, um Sperr-Modi mittels einer Ansammlung von verschiedenen Arten von Genehmigungen und Verpflichtungen zu ersetzen, der Mechanismus zum Verwalten des Zugangs zu Ressourcen erweitert. Die Genehmigungen und Verpflichtungen können zum Beispiel die Genehmigung zum Schreiben einer Ressource, zum Modifizieren einer Ressource, zum Halten einer Ressource im Zwischenspeicher, usw. sein. Spezifische Genehmigungen und Verpflichtungen werden im Folgenden in größerem Detail beschrieben.

[0049] Gemäß einem Ausführungsbeispiel sind Genehmigungen und Verpflichtungen in Genehmigungsketten kodiert. Eine Genehmigungskette kann, weil viele Genehmigungen sich eher auf eine Version einer Ressource als auf die Ressource selbst beziehen, mittels einer Ressourcen-Versions-Nummer verlängert werden. Zwei verschiedene Genehmigungsketten stehen im Konflikt, wenn sie die gleiche exklusive Genehmigung für die gleiche Version der Ressource verlangen (z. B. aktuelle Version zur Modifikation oder ein Plattenzugriff zum Schreiben). Anderenfalls sind sie kompatibel.

Gemeinschaftliches Verwenden von Genehmigungs-Transfers

[0050] Wie oben beschrieben instruiert der Hauptserver, wenn eine Ressource an einem Datenbankserver modifiziert wird und sie für weitere Modifikationen von einem anderen Datenbankserver angefordert wird, den Datenbankserver, welcher die aktuelle Kopie (CURR-Kopie) der Ressource hält, seine M-Sperre (das Recht zu modifizieren) zusammen mit der CURR-Kopie der Ressource zu dem anderen Datenbankserver weiterzugeben. Signifikanterweise wird, obgleich die Anforderung für die M-Sperre an den Hauptserver gesendet wird, das Erteilen mittels irgendeines anderen Datenbankservers (den vorherigen M-Sperren Halter) getan. Dieses Dreiecks-Benachrichtigungsystem weicht signifikant von der herkömmlichen Zweiweg-Kommunikation ab, bei welcher die Antwort auf eine Sperren-Anforderung von dem Datenbankserver erwartet wird, welche den Sperrmanager enthält, zu welchen die Sperren-Anforderung anfänglich gesendet wurde.

[0051] Gemäß einem Ausführungsbeispiel der Erfindung benachrichtigt Datenbankserver A den Hauptserver, dass die M-Sperre übertragen wurde, wenn der Halter der CURR Kopie der Ressource (z. B. Datenbankserver A) die M-Sperre an einen anderen Datenbankserver weiterreicht. Jedoch wartet der

Datenbankserver A nicht auf Bestätigung, dass der Hauptserver die Benachrichtigung erhalten hat, sondern sendet die CURR-Kopie und die M-Sperre vor Erhalt einer solchen Bestätigung. Mittels des nicht Abwartens erlegt die Schleifenkommunikation zwischen dem Hauptserver und Datenbankserver A der Übertragung keine Verzögerung auf, wodurch sich eine beträchtliche Einsparung von Protokoll-Wartezeiten ergibt.

[0052] Da Genehmigungen direkt vom aktuellen Halter der Genehmigung zu dem Anforderer der Genehmigung übertragen werden, weiß der Hauptserver nicht immer das exakte globale Bild der Sperren Erteilungen. Vielmehr weiß der Hauptserver nur von der Trajektorie der M-Sperre, von den Datenbankservern, welche „sie gerade vor Kurzem hielten", aber nicht von dem exakten Standort der Sperre zu jeder gegebenen Zeit. Gemäß einem Ausführungsbeispiel ist dieses „träge" Benachrichtigungsschema auf die M-Sperren anwendbar, aber nicht auf die W-, X- oder S-Sperren (oder ihren Entsprechungen). Verschiedenartige Ausführungsbeispiele eines Sperrenschemas sind im Folgenden in größerem Detail beschrieben.

Absturz-Wiederherstellung

[0053] Innerhalb der Kontexts der Erfindung wird davon gesprochen, dass ein Datenbankserver abgestürzt ist, wenn ein Zwischenspeicher, welcher mit dem Server assoziiert ist, unzugänglich wird. Datenbanksysteme, welche das direkte Zwischen-Server-Verschicken von unreinen Ressourcen mittels Verwendens der hierin beschriebenen Techniken anwenden, vermeiden die Notwendigkeit des Zusammenfügens von Wiederherstellungsprotokollen in Erwiderung auf einen Einzel-Server-Absturz. Gemäß einem Ausführungsbeispiel werden Einzel-Server-Abstürze wie in Fig. 4 erläutert gehandhabt. Bezugnehmend auf Fig. 4 führt das Wiederherstellungsverfahren, nach einem Einzel-Datenbankserver Absturz, für jede Ressource, welche in dem Zwischenspeicher des abgestürzten Datenbankservers gehalten wurde, das Folgende aus:
(Schritt **400**) Bestimmen des Datenbankservers, welcher die aktuellste Version der Ressource hält;
(Schritt **402**) wenn der Datenbankserver, welcher in Schritt **400** bestimmt wurde, nicht der abgestürzte Datenbankserver ist, dann (Schritt **404**) schreibt der bestimmte Datenbankserver seine zwischengespeicherte Version der Ressource auf Platte und (Schritt **406**) alle PI-Versionen der Ressource werden freigegeben. Diese Version wird alle an der Ressource durchgeführten Änderungen haben (inklusive derer, welche mittels des abgestürzten Datenbankservers gemacht wurden) und daher braucht kein Wiederherstellungsprotokoll irgendeines Datenbankservers angewandt werden.

[0054] Wenn der Datenbankserver, welcher in Schritt **402** bestimmt wurde, der abgestürzte Datenbankserver ist, dann (Schritt **408**) schreibt der Datenbankserver, welcher die letzte PI-Version der Ressource hält, seine zwischengespeicherte Version der Ressource auf Platte heraus und (Schritt **410**) alle früheren PI-Versionen werden freigegeben. Die Version, welche auf Platte herausgeschrieben wurde, wird die begangenen Änderungen haben, welche von allen Datenbankservern außer dem abgestürzten Datenbankserver an der Ressource durchgeführt wurden. Das Wiederherstellungsprotokoll des abgestürzten Datenbankservers wird angewandt (Schritt **412**), um die begangenen Änderungen, welche mittels des abgestürzten Datenbankservers durchgeführt wurden, wiederherzustellen.

[0055] Alternativ kann die letzte PI-Version der Ressource als Startpunkt zur Wiederherstellung der aktuellen Version eher im Zwischenspeicher als auf Platte verwendet werden. Insbesondere können die entsprechenden Aufzeichnungen des Wiederherstellungsprotokolls des abgestürzten Datenbankservers direkt auf die letzte PI-Version, welche im Zwischenspeicher liegt, angewandt werden, wodurch die aktuelle Version im Zwischenspeicher des Datenbankservers, welcher die letzte PI-Version hält, rekonstruiert wird.

Mehrfach-Datenbankserver-Absturz

[0056] Im Falle eines Mehrfach-Server-Absturzes, wenn weder die letzte PI-Kopie noch irgendeine CURR Kopie überlebt hat, kann es passieren, dass die Änderungen, welche an der Ressourcen vorgenommen wurden, über mehrere Protokolle der abgestürzten Datenbankserver verstreut sind. Unter diesen Umständen müssen die Protokolle der abgestürzten Datenbankserver zusammengefügt werden. Jedoch müssen nur die Protokolle der abgestürzten Datenbankserver zusammengefügt werden und nicht Protokolle von allen Datenbankservern. Dadurch ist der Umfang der Arbeit, welche für das Wiederherstellen benötigt wird, proportional zu dem Ausmaß des Absturzes und nicht zu der Größe der gesamten Konfiguration.

[0057] In Systemen, in denen es möglich ist zu bestimmen, welche abgestürzten Datenbankserver die Ressource aktualisiert haben, müssen nur die Protokolle der abgestürzten Datenbankserver, welche die Ressource aktualisiert haben, zusammengefügt und angewandt werden. Gleichermaßen brauchen in Systemen, in denen es möglich ist, zu bestimmen, welche der abgestürzten Datenbankservern die Ressource nach der dauerhaft gespeicherten Version der Ressource aktualisiert haben, nur die Protokolle der abgestürzten Datenbankserver, welche die Ressource nach der dauerhaft gespeicherten Version der Ressource aktualisiert haben, zusammengefügt und

angewandt werden.

Beispielhafter Betrieb

[0058] Zum Zwecke der Erläuterung soll im Bezug auf Fig. 1 eine exemplarische Serie von Ressourcen Übertragungen beschrieben werden. Während der Serie von Übertragungen wird an mehreren Datenbankservern auf eine Ressource zugegriffen. Insbesondere wird die Ressource zum Modifizieren entlang eines Clusterknotens verschickt und dann verursacht ein Prüfpunkt an einem der Datenbankserver einen physischen E/A dieser Ressource.

[0059] Wieder auf die Fig. 1 bezugnehmend gibt es vier Datenbankserver: A, B, C und D. Datenbankserver D ist der Hauptserver der Ressource. Datenbankserver C modifiziert die Ressource zuerst. Datenbankserver C hat die Ressourcen Version **8**. An diesem Punkt, hat Datenbankserver C auch eine M-Sperre (ein exklusives Modifikationsrecht) an dieser Ressource.

[0060] Angenommen, dass zu diesem Punkt Datenbankserver B die Ressource, welche derzeit Datenbankserver C hält, modifizieren möchte. Datenbankserver B sendet eine Anforderung (1) für eine M-Sperre an der Ressource. Datenbankserver D reiht die Anforderung in eine Modifizierer-Warteschlange ein, welche mit der Ressource assoziiert ist, und instruiert (Nachricht 2: BAST) Datenbankserver C zum:

(a) Weiterreichen der Modifikations-Genehmigung (M-Sperre) an Datenbankserver B,
(b) Senden des aktuellen Abbildes der Ressource an Datenbankserver B, und
(c) Herabsetzen der M-Sperre des Datenbankservers C auf eine H-Sperre

[0061] Nach dieser Herabsetzungs-Operation ist C verpflichtet, seine Version der Ressource (die PI-Kopie) in seinem Puffer-Zwischenspeicher zu halten.

[0062] Datenbankserver C führt die angeforderten Operationen durch und kann zusätzlich das Protokollieren der neuen Änderungen erzwingen. Zusätzlich benachrichtigt Datenbankserver C „träge" (3 AckM) den Hauptserver, dass er die Operationen (AST) durchgeführt hat. Die Benachrichtigung informiert den Hauptserver auch, dass Datenbankserver C die Version **8** hält. Datenbankserver C wartet nicht auf irgendeine Bestätigung vom Hauptserver. Als Konsequenz daraus ist es möglich, dass Datenbankserver B eine M-Sperre erhält, bevor der Hauptserver davon weiß.

[0063] Indessen wird angenommen, dass Datenbankserver A auch entscheidet, die Ressource zu modifizieren. Datenbankserver A sendet eine Nachricht (4) zu Datenbankserver D. Diese Nachricht kann vor der asynchronen Benachrichtigung vom Datenbankserver C bei Datenbankserver D ankommen.

[0064] Datenbankserver D (der Hauptserver) sendet eine Nachricht (5) zu Datenbankserver B, den letzten bekannten Modifizierer dieser Ressource, die Ressource (nachdem B sie bekommen und modifiziert hat) an Datenbankserver A weiterzureichen. Zu beachten ist, dass Datenbankserver D nicht weiß, ob die Ressource dort ist oder noch nicht. Aber Datenbankserver D weiß, dass die Ressource schließlich bei B ankommen wird.

[0065] Nachdem Datenbankserver B die Ressource bekommen hat und die beabsichtigten Änderungen durchgeführt hat (jetzt hat B die Version **9** der Ressource), setzt er seine eigene Sperre auf H herab, sendet (6) zusammen mit der M-Sperre die aktuelle Version der Ressource („CURR-Ressource") zu Datenbankserver A. Datenbankserver B sendet auch eine „träge" Benachrichtigung (6 AckM) an den Hauptserver.

[0066] Während diese Ressource im Datenbankserver A modifiziert wird, wird angenommen, dass ein Prüfpunktmechanismus an Datenbankserver C entscheidet, die Ressource auf Platte zu schreiben. Hinsichtlich der oben beschriebenen asynchronen Ereignisse, wird angenommen, dass 3 AckM und 6 AckM schon beim Hauptserver angekommen sind. Die Operationen, welche in Erwiderung auf die mit Prüfpunkt versehende Operation ausgeführt werden, werden unter Bezug auf Fig. 5 erläutert.

[0067] Bezugnehmend auf Fig. 5 sendet, weil der Datenbankserver C eine H-Sperre, welche kein Schreibprivileg beinhaltet, auf Version **8** hält, Datenbankserver C Nachricht 1 an den Hauptserver (D), womit er die W- (Schreib-) Sperre für seine Version anfordert. Zu diesen Zeitpunkt weiß der Hauptserver, dass die Ressource zu Datenbankserver A geschickt wurde (angenommen, dass die Benachrichtigung angekommen ist). Datenbankserver D sendet eine (unaufgeforderte) W-Sperre an Datenbankserver A (2 BastW) mit der Instruktion die Ressource zu schreiben.

[0068] Im allgemeinen Fall wird diese Instruktion zu dem letzten Datenbankserver gesendet, dessen gesendete Benachrichtigung angekommen ist (oder zu dem Datenbankserver, welcher die Ressource vom letzten bekannten Sender erhalten soll). Datenbankserver A schreibt (3) seine Version der Ressource. Die Ressource, welche vom Datenbankserver A geschrieben wird, ist Version **10** der Ressource. Zu dieser Zeit kann die aktuelle Kopie der Ressource irgendwo anders sein, wenn zusätzliche Anforderer die Ressource verlangten. Die Platte bestätigt, wenn das Schreiben erledigt ist (4 Ack).

[0069] Wenn das Schreiben abschließt, versorgt Datenbankserver A Datenbankserver D mit der Information, dass Version **10** jetzt auf Platte ist (5 AckW). Datenbankserver A setzt freiwillig seine W-Sperre herab (wofür er nicht zuerst um Erlaubnis fragt).

[0070] Der Hauptserver (D) richtet sich an den Datenbankserver C und, anstelle des Erteilens der angeforderten W-Sperre, benachrichtigt C, dass das Schreiben komplettiert ist (6). Der Hauptserver teilt die aktuelle Versions-Nummer auf der Platte an die Halter aller PI-Kopien mit, so dass alle früheren PI-Kopien auf C freigegeben werden können. In diesem Szenario setzt es, da Datenbankserver C keine PI-Kopien älter als 10 hat, die Sperre des Datenbankservers C auf NULL.

[0071] Der Hauptserver sendet auch eine Bestätigungsnachricht zu Datenbankserver B, wobei er Datenbankserver B instruiert seine PI-Kopien, welche früher als 10 sind, freizugeben (7 AckW(10)).

Der Verteilte-Sperrmanager

[0072] Im Gegensatz zur konventioneller DLM Logik, kann der Hauptserver in einem System, welches die Direkt-Verschickungstechniken, welche hierin beschrieben sind, realisiert, unvollständige Informationen über die Zustände der Sperren an den Datenbankservern haben. Gemäß einem Ausführungsbeispiel unterhält der Hauptserver einer Ressource die folgenden Informationen und Datenstrukturen:

(1) eine Warteschlange von CURR-Kopie Anforderern (entweder für Modifikationen oder für gemeinsamen Zugriff) (die obere Grenze der Warteschlangenlänge ist die Anzahl der Datenbankserver in dem Cluster). Diese Warteschlange wird hierin als Aktuelle-Anforderungs-Warteschlange (CQ) bezeichnet.

(2) wenn eine Ressource an einen anderen CURR Anforderer gesendet wird, benachrichtigen die Sender träge (asynchron im Sinne, dass diese nicht auf eine Bestätigung warten) den Hauptserver über das Ereignis. Der Hauptserver führt Buch über die letzten paar Sender. Dies ist ein Pointer auf die CQ.

(3) Die Versionsnummer der letzten Ressourcenversion auf Platte.

(4) W-Sperren Erteilungen und eine W-Anforderungen-Warteschlange.

[0073] Gemäß einem Ausführungsbeispiel ist W-Genehmigung synchron: Sie wird nur vom Hauptserver erteilt und der Hauptserver stellt sicher, dass es für diese Ressource nicht mehr als einen Schreiber in dem Cluster gibt. Der Hauptserver kann die nächste Erteilung nur machen, nachdem er benachrichtigt wurde, dass das vorherige Schreiben komplettiert und die W-Sperre freigegeben wurde. Wenn es mehr als einen Modifizierer gibt, wird eine W-Sperre für die Zeitdauer des Schreibens gegeben und freiwillig nach dem Schreiben freigegeben. Wenn da nur ein Modifizierer ist, kann der Modifizierer die W-Genehmigung behalten.

(5) eine Liste von H-Sperren Haltern mit ihren entsprechenden Ressourcen Versions-Nummern. Dies stellt Informationen (obgleich möglicherweise unvollständig) über die PI-Kopien in Puffer-Zwischenspeichern bereit.

Plattenerneuern

[0074] Da die direkten Versendetechniken, welche hierin beschrieben sind, signifikant die Lebensdauern der Puffer-Zwischenspeicher-Abbilder der Ressourcen und der Platten-Abbilder trennt, gibt es eine Notwendigkeit, diese Wiederherstellungslücke zu überbrücken. Gemäß einem Ausführungsbeispiel wird ein neuer Schritt von Wiederherstellung zwischen DLM-Wiederherstellung und Puffer-Zwischenspeicher-Wiederherstellung hinzugefügt. Dieser neue Wiederherstellungsschritt wird hierein als „Plattenerneuern" bezeichnet.

[0075] Obwohl während normaler Zwischenspeicher-Operationen ein Hauptserver einer Ressource nur ungefähres Wissen über den Ort der Ressource und über die Verfügbarkeit von PI- und CURR-Kopien hat, sammelt bei DLM-Wiederherstellung (welche Zwischenspeicher-Wiederherstellung vorangeht) der Hauptserver einer Ressource komplette Informationen über die Verfügbarkeit der letzten PI- und CURR-Kopien in den Puffer-Zwischenspeichern überlebender Datenbankserver. Dies gilt egal, ob oder nicht der Hauptserver der Ressource ein neuer Hauptserver (wenn vor dem Absturz die Ressource von einem abgestürzten Datenbankserver gehandhabt wurde) oder ein überlebender Hauptserver ist.

[0076] Nach Sammeln dieser Informationen weiß der Hauptserver, welcher Datenbankserver die letzte Kopie der Ressource besitzt. In dem Zustand des „Plattenerneuerns", gibt der Hauptserver eine W-Sperre an den Besitzer dieser letzten Kopie der Ressource heraus (CURR wenn sie verfügbar ist und letzte PI-Kopie wenn die CURR-Kopie zusammen mit dem abgestürzten Datenbankserver verschwunden ist). Der Hauptserver instruiert dann diesen Datenbankserver, die Ressource auf Platte zu schreiben. Wenn das Schreiben komplettiert ist, wandeln alle anderen Datenbankserver ihre H-Sperren in NULL-Sperren um (weil die geschriebene Kopie die letzte Verfügbare ist). Nachdem diese Sperren umgewandelt wurden, kann Zwischenspeicher-Wiederherstellung wie normal weitergehen.

[0077] Einige Optimierungen sind während des Zustands des Plattenerneuerns möglich. Zum Beispiel muss die Ressource nicht notwendigerweise auf Platte geschrieben werden, wenn das letzte Abbild in

dem Puffer-Zwischenspeicher des Datenbankservers ist, welcher das Wiederherstellen durchführt.

Alternativen zu Sperren-basiertem Schema

[0078] Verschiedenartige Techniken für direktes Verschicken unreiner Kopien von Ressourcen zwischen Datenbankservern wurden im Zusammenhang mit einem Sperren-Schema beschrieben, welches spezielle Typen von Sperren (M-, W-, und H-Sperren) verwendet.

[0079] Insbesondere werden diese Spezialsperren verwendet, um sicherzustellen, dass (1) nur der Server mit der aktuellen Version der Ressource die Ressource modifiziert, (2) alle Server ihre PI-Versionen der Ressource halten, bis dieselbe Version oder eine neuere Version der Ressource auf Platte geschrieben ist und (3) die plattenresidente Version der Ressource nicht von einer älteren Version der Ressource überschrieben wird.

[0080] Jedoch ist das sperrenbasierte Zugriffs-Steuerschema bloß ein Kontext, in welchem diese Erfindung realisiert werden kann. Zum Beispiel können dieselben drei Regeln mittels irgendeiner Variation von Zugriffs-Steuerungsschemen durchgesetzt werden. Daher ist diese Erfindung nicht auf irgendein besonderes Zugriffs-Steuerungsschema beschränkt.

[0081] Zum Beispiel kann Zugriff, eher als durch ein auf Sperren basierenden Verwalten des Zugriffes auf eine Ressource, mittels Markern verwaltet werden, wobei jeder Marker einen besonderen Typ von Genehmigung repräsentiert. Die Marker für eine bestimmte Ressource können zwischen den Parallel-Servern in einer Weise übertragen werden, welche sicherstellt, dass die drei oben gegebenen Regeln durchgesetzt werden.

[0082] Gleichermaßen können die Regeln mittels Verwendens eines Zustands-basierten Schemas durchgesetzt werden. In einem Zustands-basierten Schema ändert eine Version einer Ressource den Zustand in Erwiderung auf Ereignisse, wo der Zustand einer Version den Typ von Aktionen diktiert, welche an dieser Version durchgeführt werden können. Zum Beispiel empfängt ein Datenbankserver die aktuelle Version einer Ressource in ihrem „aktuellen" Zustand. Der aktuelle Zustand erlaubt Modifikation der Ressource und das Auf-Platte-Schreiben der Ressource. Wenn ein Datenbankserver die aktuelle Version der Ressource zu einem anderen Knoten überträgt, ändert sich die gehaltene Version in einen „PI schreibbaren" Zustand. In dem PI schreibbaren Zustand kann die Version (1) nicht modifiziert werden, (2) kann nicht überschrieben werden, aber (3) kann auf Platte geschrieben werden. Wenn irgendeine Version der Ressource auf Platte geschrieben wird, werden alle Versionen, welche in einem PI-schreibbaren-Zustand sind, welche gleich oder älter als die Version sind, welche auf Platte geschrieben wurde, in einem „PI-freigegebenen"-Zustand gesetzt. In dem PI-freigegebenen-Zustand können Versionen überschrieben werden, aber sie können nicht modifiziert oder auf Platte geschrieben werden.

Hardwareübersicht

[0083] Fig. 6 ist ein Blockdiagramm, welches ein Computersystem **600** erläutert, an welchem ein Ausführungsbeispiel der Erfindung realisiert werden kann. Computersystem **600** enthält einen Bus **602** oder einen anderen Kommunikationsmechanismus zum Kommunizieren von Information und einen Prozessor **604**, welcher zum Prozessieren von Informationen mit dem Bus **602** gekoppelt ist. Computersystem **600** weist auch einen Hauptspeicher **606**, wie zum Beispiel einen Direktzugriffsspeicher (RAM) oder eine andere mit dem Bus **602** gekoppelte dynamische Speichervorrichtung auf, zum Speichern von Informationen und Instruktionen, welche mittels des Prozessors **604** ausgeführt werden müssen. Hauptspeicher **606** kann auch zum Speichern von temporären Variablen oder anderen Zwischeninformationen während des Ausführens von Instruktionen verwendet werden, welche mittels des Prozessors **604** ausgeführt werden müssen. Computersystem **600** enthält ferner einen Festwertspeicher (ROM) **608** oder eine andere mit Bus **602** gekoppelte statische Speichervorrichtung zum Speichern statischer Informationen und Instruktionen für den Prozessor **604**. Eine Speichervorrichtung **610**, wie zum Beispiel eine Magnetplatte oder optische Platte ist bereitgestellt und mit dem Bus **602** gekoppelt, zum Speichern von Informationen und Instruktionen.

[0084] Computersystem **600** kann zum Anzeigen von Informationen für einen Benutzer mittels des Bus **602** mit einer Anzeige **612**, wie zum Beispiel einer Kathodenstrahlröhre (CRT), gekoppelt werden. Eine Eingabevorrichtung **614**, welche alphanumerische und andere Tasten aufweist, ist zum Kommunizieren von Informationen und Befehlsauswahlen zu den Prozessor **604**, an den Bus **602** gekoppelt. Eine andere Art von Benutzer-Eingabevorrichtung ist Kursorsteuerung **616** wie zum Beispiel eine Maus, ein Trackball oder Kursor-Richtungs-Tasten zum Kommunizieren von Richtungsinformationen und Befehlsauswahlen an den Prozessor **604** und zum Steuern von der Kursorbewegung auf einem Display **612**. Diese Eingabevorrichtung hat typischerweise zwei Freiheitsgrade entlang zweier Achsen, einer ersten Achse (z. B. x) und einer zweiten Achse (z. B. y), welches der Vorrichtung erlaubt, Positionen in einer Ebene zu spezifizieren.

[0085] Die Erfindung betrifft das Benutzen eines Computersystems **600** zum Reduzieren des Over-

heads, welcher mit einem Ping assoziiert ist. Gemäß einem Ausführungsbeispiel der Erfindung wird der mit einem Ping assoziierte Overhead, mittels des Computersystems **600** in Erwiderung auf den Prozessor **604**, der eine oder mehrere Sequenzen einer oder mehrerer Instruktionen ausführt, welche im Hauptspeicher **606** enthalten sind, reduziert. Solche Instruktionen können von einem anderen computerlesbaren Medium, wie zum Beispiel einer Speichervorrichtung **610**, in den Hauptspeicher **606** eingelesen werden. Ausführen der Sequenzen von Instruktionen, welche im Hauptspeicher **606** enthalten sind, veranlassen den Prozessor **604**, die Prozessschritte, welche hierin beschrieben sind, auszuführen. In alternativen Ausführungsbeispielen können hartverdrahtete Schaltkreise anstelle von oder in Kombination mit Softwareinstruktionen verwendet werden, um die Erfindung zu realisieren. Daher sind die Ausführungsbeispiele der Erfindung nicht auf irgendeine spezifische Kombination von Hardware-Schaltkreisen und Software begrenzt.

[0086] Der Ausdruck „Computer-lesbares Medium" wie er hierin verwendet wird, bezieht sich auf jedes Medium, welches bei dem Bereitstellen von Instruktionen zum Ausführen an den Prozessor **604** mitwirkt. Solch ein Medium kann viele Formen annehmen, welche einschließen aber nicht begrenzt sind auf nichtflüchtige Medien, flüchtige Medien und Übertragungsmedien. Nichtflüchtige Medien schließen zum Beispiel optische oder magnetische Platten wie beispielsweise die Speichervorrichtung **610** ein. Flüchtige Medien schließen dynamischen Speicher, wie zum Beispiel den Hauptspeicher **606** ein. Übertragungsmedien schließen Koaxialkabel, Kupferdraht und Lichtwellenleiter, inklusive der Drähte, welche der Bus **602** aufweist, ein. Übertragungsmedien können ferner die Form von akustischen oder Lichtwellen annehmen, wie zum Beispiel solche, die während Radiowellen- und Infrarot-Datenkommunikationen erzeugt werden.

[0087] Übliche Formen von Computer-lesbaren Medien schließen zum Beispiel eine Floppy-Disk, eine flexible Diskette, Festplatte, Magnetband oder jedes andere magnetische Medium, eine CD-ROM, jedes andere optische Medium, Lochkarten, Lochstreifen, jedes andere physische Medium mit Mustern von Löchern, ein RAM, ein PROM und EPROM, ein FLASH-EPROM, jeden anderen Speicherchip oder Speicherkassette, eine Trägerwelle wie nachfolgend hierin beschrieben oder jedes andere Medium, von welchem ein Computer lesen kann, ein.

[0088] Verschiedenartige Formen von Computer-lesbaren Medien können in das Tragen einer oder mehrerer Sequenzen von einer oder mehrerer Instruktionen zum Prozessor **604** zum Ausführen einbezogen sein. Zum Beispiel können die Instruktionen anfänglich auf einer Magnetplatte eines abseitigen Computers gespeichert sein. Der Fern-Computer kann die Instruktionen in seinen dynamischen Speicher laden und die Instruktionen mittels Verwendens eines Modems über eine Telefonleitung versenden. Ein Modem am Ort des Computersystems **600** kann die Daten über die Telefonleitung empfangen und einen Infrarot-Transmitter verwenden, um die Daten in ein Infrarotsignal umzuwandeln. Ein Infrarotdetektor kann die Daten, welche in dem Infrarotsignal enthalten sind, empfangen und ein passender Schaltkreis kann die Daten auf den Bus **602** platzieren. Bus **602** trägt die Daten zum Hauptspeicher **606**, von welchem Prozessor **604** die Instruktionen abruft und ausführt. Die mittels des Hauptspeichers **606** empfangenen Instruktionen können, entweder vor oder nach dem Ausführen mittels des Prozessors **604** optional in einer Speichervorrichtung **610**, gespeichert werden.

[0089] Computersystem **600** gehört zu einem Gemeinschafts-Plattensystem, in welchem Daten eines oder mehrerer Speichervorrichtungen (z. B. Plattenlaufwerke **655**) für beide, Computersystem **600** und für eine oder mehrere CPUs (z. B. CPU **651**), zugreifbar sind. In dem erläuterten System wird der Gemeinschaftszugriff auf die Plattenlaufwerke **655** mittels eines Systemnetz-Bereiches **653** bereitgestellt. Jedoch können verschiedenartige Mechanismen alternativ genutzt werden, um Gemeinschaftszugriff bereitzustellen.

[0090] Computersystem **600** weist auch eine an den Bus **602** gekoppelte Kommunikations-Schnittstelle **618** auf. Die Kommunikations-Schnittstelle **618** stellt eine bidirektionale Daten-Kommunikationsverbindung zu einer Netzwerk-Verbindung **620** bereit, welche mit einem lokalen Netzwerk **622** gekoppelt ist. Zum Beispiel kann Kommunikations-Schnittstelle **618** eine dienstintegrierendes-digitales-Nachrichtennetz- (ISDN) Karte oder ein Modem sein, um eine Datenkommunikationsverbindung zu einer entsprechenden Ausführung von Telefonleitung bereitzustellen. Als ein anderes Beispiel kann die Kommunikations-Schnittstelle **618** eine Lokales-Datennetz- (LAN) Karte sein, um eine Datenkommunikationsverbindung zu einem kompatiblen LAN bereitzustellen. Ebenso können drahtlose Verbindungen realisiert werden. In einer solchen Realisierung, sendet und empfängt Kommunikations-Schnittstelle **618** elektrische, elektromagnetische oder optische Signale, welche digitale Datenströme tragen, welche verschiedenartige Typen von Informationen repräsentieren.

[0091] Netzwerk-Verbindung **620** stellt typischerweise mittels eines oder mehrerer Netzwerke Datenkommunikation zu anderen Datenvorrichtungen bereit. Zum Beispiel kann die Netzwerk-Verbindung **620** mittels des lokalen Netzwerks **622** eine Verbindung zu einem Zentralrechner **624** oder zu Datenvorrich-

tungen, welche von einem Internet-Service-Provider (ISP) **626** betrieben werden, bereitstellen. Im Gegenzug stellt ISP **626** Datenkommunikationsservices mittels des Weltweiten-Datenpaket-Kommunikations-Netzwerkes, jetzt gemeinhin als „Internet" **628** bezeichnet, bereit. Das Lokale Netzwerk **622** und Internet **628** nutzen beide elektrische, elektromagnetische oder optische Signale, welche digitale Datenströme tragen. Die Signale durch die verschiedenartigen Netzwerke hindurch und die Signale an der Netzwerk-Verbindung **620** und durch Kommunikations-Schnittstelle **618** hindurch, welche die digitalen Daten zu und vom Computersystem **600** tragen, sind beispielhafte Arten von Trägerwellen, welche die Informationen tragen.

[0092] Computersystem **600** kann durch das/die Netzwerk(e), Netzwerk-Verbindung **620** und Kommunikations-Schnittstelle **618** hindurch Nachrichten senden und Daten, inklusive Programmcode, empfangen. In dem Internetbeispiel kann ein Server **630** mittels Internet **628**, ISP **626**, lokales Netzwerk **622** und Kommunikations-Schnittstelle **618** einen angeforderten Code für ein Anwendungsprogramm übertragen.

[0093] Der empfangene Code kann mittels des Prozessors **604** so wie er empfangen wurde, ausgeführt werden, und/oder in der Speichervorrichtung **610** oder anderen nichtflüchtigen Speichern zum späteren Ausführen gespeichert werden. Auf diese Weise kann das Computersystem **600** Anwendungscode in Form einer Trägerwelle erhalten.

[0094] Während Techniken zum Handhaben von Pings hierin in Bezug auf Pings beschrieben wurden, welche auftreten, wenn Mehrfach-Datenbankserver auf eine gemeinsame, Dauerhaft-Speichervorrichtung Zugriff haben, sind diese Techniken nicht auf diesen Kontext beschränkt. Insbesondere können diese Techniken in jedem Umfeld angewandt werden, in welchem ein Prozess, welcher mit einem Zwischenspeicher assoziiert ist, eine Ressource anfordert, deren aktuelle Version sich in einem anderen Zwischenspeicher befindet. Solche Umfelder schließen zum Beispiel Umfelder, in welchen Textserver an verschiedenen Knoten auf das selbe Textmaterial Zugriff haben, Umfelder, in welchen Medienserver an verschiedenen Knoten Zugriff auf die selben Videodaten haben, usw. ein.

[0095] Handhaben von Pings mittels der hierin beschriebenen Techniken, stellt effektives Zwischen-Datenbankserver-Übertragen von Ressourcen bereit, so dass die Betriebszeit-Leistungsfähigkeit gut mit der ansteigenden Anzahl von Datenbankservern und Benutzern je Datenbankserver skaliert. Zusätzlich laufen die Techniken auf eine effiziente Wiederherstellung von Einzel-Datenbankserver-Abstürzen (dem häufigsten Typ von Abstürzen) hinaus, welche gut mit der anwachsenden Zahl von Datenbankservern skaliert.

[0096] Signifikanterweise handhaben die hierin beschriebenen Techniken Pings mittels Sendens von Ressourcen über den IPC Transport und nicht mittels Platten-Intervention. Folglich sind Platten E/As für Ressourcen, welche in einem Ping resultieren, im Wesentlichen beseitigt. Ein synchroner E/A ist nur so lange involviert, wie er für das Erzwingen des Protokolls nötig ist. Zusätzlich, obgleich Platten E/A zum Prüfpunkt setzen und Puffer-Zwischenspeichern-Austausch übernommen wird, verlangsamen solche E/A nicht das Puffer-Verschicken quer durch den Cluster.

[0097] Die direkten Versendungstechniken, welche hierin beschrieben wurden tendieren auch dazu die Anzahl von Kontext-Umschaltungen, welche durch ein Ping auftreten, zu reduzieren. Insbesondere wird die Sequenz von Schleifen-Nachrichten zwischen den Teilnehmern des Protokolls (Anforderer und Halter) und dem Hauptserver mittels des Kommunikationsdreiecks ersetzt: Anforderer, Hauptserver, Halter, Anforderer.

[0098] In der vorangegangenen Ausführung wurde die Erfindung mit Bezug auf spezifische Ausführungsbeispiele hiervon beschrieben. Es wird jedoch klar sein, dass dazu verschiedenartige Modifikationen und Änderungen gemacht werden können, ohne vom Schutzumfang der Erfindung abzuweichen. Die Ausführung und Figuren sind gemäß mehr in einem erklärenden als in einem einschränkenden Sinne zu beachten.

Patentansprüche

1. Verfahren zum Übertragen einer Ressource von einem ersten Zwischenspeicher zu einem zweiten Zwischenspeicher, wobei das Verfahren die Schritte aufweist:
Beibehalten einer ersten Kopie der Ressource in dem ersten Zwischenspeicher während des Übertragens einer zweiten Kopie der Ressource von dem ersten Zwischenspeicher zu dem zweiten Zwischenspeicher, ohne die Ressource von dem ersten Zwischenspeicher vorher dauerhaft in einen persistenten Speicher (**655**) zu speichern; und
Beibehalten mindestens einer Kopie der Ressource in dem ersten Zwischenspeicher bis die erste Kopie der Ressource oder ein Versionsnachfolger von ihr dauerhaft gespeichert ist.

2. Verfahren gemäß Anspruch 1, wobei der erste Zwischenspeicher ein Zwischenspeicher ist, der von einem ersten Datenbankserver (A, B, C, D) verwaltet wird, und der zweite Zwischenspeicher ein Zwischenspeicher ist, der von einem zweiten Datenbankserver (A, B, C, D) verwaltet wird.

3. Verfahren gemäß Anspruch 1, ferner aufweisend die Schritte:
Ermöglichen, dass die erste Kopie der Ressource in dem ersten Zwischenspeicher verändert wird, bevor die zweite Kopie zu dem zweiten Zwischenspeicher übertragen wird; und
Verhindern, dass die erste Kopie der Ressource verändert wird, nachdem die zweite Kopie zu dem zweiten Zwischenspeicher übertragen worden ist.

4. Verfahren gemäß Anspruch 1, ferner aufweisend die Schritte:
Senden einer Anforderung für die Genehmigung zum Freigeben der ersten Kopie, nachdem die zweite Kopie zu dem zweiten Zwischenspeicher übertragen worden ist;
Veranlassen, dass die erste Kopie oder ein Versionsnachfolger von ihr als Reaktion auf die Anforderung dauerhaft gespeichert wird (**302**); und
Senden einer Nachricht, die anzeigt, dass die erste Kopie freigegeben werden kann (**304**), als Reaktion darauf, dass der Versionsnachfolger dauerhaft gespeichert ist.

5. Verfahren gemäß Anspruch 4, wobei:
der Schritt des Sendens einer Anforderung für die Genehmigung zum Freigeben der ersten Kopie mittels eines Sendeprozesses durchgeführt wird; und
der Schritt des Veranlassens, dass die erste Kopie oder ein Versionsnachfolger von ihr dauerhaft gespeichert wird, den Schritt des Veranlassens, dass ein Prozess, der ein anderer als der Sendeprozess ist, einen Versionsnachfolger der ersten Kopie der Ressource speichert, aufweist (**302**).

6. Verfahren gemäß Anspruch 1, wobei der Schritt des Beibehaltens mindestens einer Kopie der Ressource in dem ersten Zwischenspeicher die Schritte aufweist:
Bestimmen, ob eine dauerhaft gespeicherte Kopie der Ressource neuer als die erste Kopie ist, bevor versucht wird die erste Kopie dauerhaft zu speichern;
Freigeben der ersten Kopie ohne dauerhaftes Speichern der ersten Kopie, falls die dauerhaft gespeicherte Kopie neuer als die erste Kopie ist; und
dauerhaftes Speichern der ersten Kopie, falls die dauerhaft gespeicherte Kopie nicht neuer als die erste Kopie ist.

7. Verfahren gemäß Anspruch 3, ferner aufweisend den Schritt des Übertragens einer Änderungsgenehmigung von einem Sendeprozess, der dem ersten Zwischenspeicher zugeordnet ist (**204**), zu einem Empfangsprozess, der dem zweiten Zwischenspeicher zugeordnet ist, zusammen mit der zweiten Kopie der Ressource.

8. Verfahren gemäß Anspruch 7, wobei:
Genehmigungen für das Zugreifen auf die Ressource mittels eines Hauptservers (D) verwaltet werden; und
der Schritt des Übertragens der Änderungsgenehmigung zu dem Empfangsprozess vor dem Empfangen der Bestätigung von dem Hauptserver (D) für die Übertragung der Änderungsgenehmigung zu dem Empfangsprozess durchgeführt wird (**204**).

9. Verfahren gemäß Anspruch 1, ferner aufweisend die Schritte:
Übertragen einer Änderungsgenehmigung von einem Sendeprozess, der dem ersten Zwischenspeicher zugeordnet ist, an einen Empfangsprozess (**204**), der dem zweiten Zwischenspeicher zugeordnet ist, zusammen mit der zweiten Kopie der Ressource;
wobei Genehmigungen für das Zugreifen auf die Ressource mittels eines Hauptservers (D) verwaltet werden; und
wobei der Schritt des Übertragens der Änderungsgenehmigung zu dem Empfangsprozess durchgeführt wird, bevor der Hauptserver (D) die Bestätigung für die Übertragung der Änderungsgenehmigung zu dem Empfangsprozess empfängt (**204**).

10. Verfahren gemäß Anspruch 1, ferner aufweisend die Schritte:
ein Empfangsprozess, der dem zweiten Zwischenspeicher zugeordnet ist, sendet eine Anforderung für die Ressource an einen Hauptserver (D) der Ressource;
der Hauptserver (D) der Ressource sendet als Reaktion auf die Anforderung des Empfangsprozesses eine Nachricht zu einem Sendeprozess, der dem ersten Zwischenspeicher zugeordnet ist; und
der Sendeprozess überträgt die zweite Kopie zu dem Empfangsprozess als Reaktion auf die Nachricht von dem Hauptserver (D).

11. Verfahren gemäß Anspruch 1, das ferner nach dem Schritt der Übertragung der zweiten Kopie zu dem zweiten Zwischenspeicher die Ausführung der folgenden Schritte aufweist:
ein Sendeprozess, der dem ersten Zwischenspeicher zugeordnet ist, fordert eine Sperre (**300**) von einem Sperrmanager an, wobei die Sperre die Genehmigung zum Schreiben der Ressource auf Platte erteilt, aber nicht die Genehmigung zur Änderung der Ressource;
der Sperrmanager wählt einen Prozess aus, der eine Version der Ressource aufweist, die mindestens so neu wie die erste Kopie ist;
der Sperrmanager erteilt dem ausgewählten Prozess die Sperre; und
der ausgewählte Prozess schreibt die Version der Ressource auf Platte.

12. Verfahren gemäß Anspruch 11, ferner aufweisend den Schritt, dass der Sperrmanager als Reaktion auf das Schreiben der Version der Ressource auf Platte (**302**), veranlasst, dass alle Versionen der Ressource, die älter als die Version sind, freigegeben

werden (**306**).

13. Verfahren gemäß Anspruch 1, das ferner, nach einem Ausfall eines Zwischenspeichers, der eine verschmutzte Kopie der Ressource enthält, die Schritte aufweist:
Bestimmen, ob der ausgefallene Zwischenspeicher die letzte Version der Ressource enthalten hat (**402**);
falls der ausgefallene Zwischenspeicher die letzte Version der Ressource enthalten hat, dann:
Schreiben eines letzten früheren Abbildes der Ressource auf Platte (**408**);
Freigeben aller vorherigen früheren Abbilder der Ressource (**410**); und
Verwenden eines Wiederherstellungsprotokolls des ausgefallenen Zwischenspeichers, um die letzte Version der Ressource zu rekonstruieren (**412**).

14. Verfahren gemäß Anspruch 13, ferner aufweisend die Schritte:
falls der ausgefallene Zwischenspeicher nicht die letzte Version der Ressource enthalten hat, dann Schreiben der letzten Version der Ressource auf die Platte (**404**); und
Freigeben aller früheren Abbild-Kopien der Ressource (**406**).

15. Verfahren gemäß Anspruch 1, das ferner, nach einem Ausfall einer Mehrzahl von Zwischenspeichern, die verschmutzte Versionen der Ressource enthalten, die Schritte aufweist:
Bestimmen, ob einer der ausgefallenen Zwischenspeicher die letzte Version der Ressource enthalten hat (**400**); und
falls irgendeiner der ausgefallenen Zwischenspeicher die letzte Version der Ressource enthalten hat, dann:
Zusammenführen und Verwenden der Wiederherstellungsprotokolle der ausgefallenen Zwischenspeicher, um die letzte Version der Ressource zu rekonstruieren.

16. Verfahren gemäß Anspruch 1, wobei die zweite Kopie der Ressource eine verschmutzte Version der Ressource ist, wobei der erste Zwischenspeicher und der zweite Zwischenspeicher zu einer Mehrzahl von Zwischenspeichern gehören, und ferner die Schritte aufweist:
Verwalten von separaten Wiederherstellungsprotokollen für jeden Zwischenspeicher der Mehrzahl von Zwischenspeichern; und
wenn ein Zwischenspeicher der Mehrzahl von Zwischenspeichern ausfällt, Wiederherstellen des ausgefallenen Zwischenspeichers basierend auf dem Wiederherstellungsprotokoll, das dem ausgefallenen Zwischenspeicher zugeordnet ist, ohne die separaten Wiederherstellungsprotokolle der anderen Zwischenspeicher der Mehrzahl von Zwischenspeichern zu überprüfen.

17. Verfahren gemäß Anspruch 16, wobei jeder Zwischenspeicher der Mehrzahl von Zwischenspeichern ein Zwischenspeicher ist, der von einem separaten Datenbank-Server (A, B, C, D) einer Mehrzahl von Datenbank-Servern (A, B, C, D) verwaltet wird.

18. Verfahren gemäß Anspruch 16, wobei das Verfahren ferner die Schritte aufweist:
Ermöglichen, dass die erste Kopie der Ressource in dem ersten Zwischenspeicher verändert wird, bevor die zweite Kopie zu dem zweiten Zwischenspeicher übertragen wird; und
Verhindern, dass die erste Kopie der Ressource verändert wird, nachdem die zweite Kopie zu dem zweiten Zwischenspeicher übertragen worden ist.

19. Verfahren gemäß Anspruch 1, wobei sich der erste Zwischenspeicher auf einem ersten Knoten einer Mehrzahl von Knoten (A, B, C, D) befindet und der zweite Zwischenspeicher sich auf einem zweiten Knoten der Mehrzahl von Knoten (A, B, C, D) befindet, und ferner die Schritte aufweist:
Verändern der Ressource in dem ersten Zwischenspeicher des ersten Knotens einer Mehrzahl von Knoten (A, B, C, D), um eine geänderte Version der Ressource zu erzeugen;
Verwalten eines Prüfpunkts für den ersten Knoten, der anzeigt, wo mit der Arbeit begonnen werden muss, wenn der erste Knoten ausfällt;
Beibehalten einer ersten Kopie der geänderten Version in dem ersten Zwischenspeicher während des Übertragens einer zweiten Kopie der geänderten Version von dem ersten Zwischenspeicher zu dem zweiten Zwischenspeicher des zweiten Knotens der Mehrzahl von Knoten (A, B, C, D), ohne die geänderte Version von dem ersten Zwischenspeicher vorher dauerhaft in einen persistenten Speicher (**655**) zu speichern; und
Vorversetzen des Kontrollpunkts als Reaktion auf einen Hinweis, dass ein anderer Knoten der Mehrzahl von Knoten (A, B, C, D) eine Version der Ressource dauerhaft gespeichert hat, die mindestens so neu wie die geänderte Version ist.

20. Verfahren gemäß Anspruch 19, ferner aufweisend den Schritt:
Beibehalten mindestens einer Kopie der geänderten Version in dem ersten Zwischenspeicher bis die erste Kopie der geänderten Version oder ein Versionsnachfolger von ihr dauerhaft gespeichert ist.

21. Verfahren gemäß Anspruch 19, ferner aufweisend den Schritt:
Freigeben der Ressource an dem ersten Knoten (**306**) als Reaktion auf den Hinweis, dass ein anderer Knoten der Mehrzahl von Knoten die Version der Ressource dauerhaft gespeichert hat, welche Version mindestens so neu wie die geänderte Version ist.

22. Computerlesbares Medium, das eine Folge

oder mehrere Folgen von Befehlen für die Übertragung einer Ressource von einem ersten Zwischenspeicher zu einem zweiten Zwischenspeicher trägt, wobei die Ausführung der einen Folge oder der mehreren Folgen von Befehlen mittels eines Prozessors oder mehrerer Prozessoren den einen Prozessor oder die mehreren Prozessoren veranlasst, die Schritte des in irgendeinem der Ansprüche 1–21 geschilderten Verfahrens durchzuführen.

23. System zum Übertragen einer Ressource, wobei das System aufweist:
einen ersten Knoten, der einen ersten Zwischenspeicher aufweist, der kommunikativ mit einem zweiten Zwischenspeicher aus einem oder mehreren anderen Zwischenspeichern, die in einem oder mehreren anderen Knoten enthalten sind, gekoppelt ist;
wobei der erste Knoten eingerichtet ist, eine erste Kopie der Ressource in dem ersten Zwischenspeicher beizubehalten, während des Übertragens einer zweiten Kopie der Ressource von dem ersten Zwischenspeicher zu dem zweiten Zwischenspeicher, ohne vorher die Ressource von dem ersten Zwischenspeicher dauerhaft in einen persistenten Speicher (**655**) zu speichern; und
wobei der erste Knoten eingerichtet ist, mindestens eine Kopie der Ressource in dem ersten Zwischenspeicher beizubehalten bis die erste Kopie der Ressource oder ein Versionsnachfolger von ihr dauerhaft gespeichert ist.

24. System gemäß Anspruch 23, wobei der erste Knoten ein erster Datenbank-Server (A, B, C, D) ist und mindestens einer der ein oder mehreren anderen Knoten ein zweiter Datenbank-Server (A, B, C, D) ist, der den zweiten Zwischenspeicher aufweist.

25. System gemäß Anspruch 23, wobei
der erste Knoten eingerichtet ist, es zu ermöglichen, dass die erste Kopie der Ressource in dem ersten Zwischenspeicher geändert wird, bevor die zweite Kopie zu dem zweiten Zwischenspeicher übertragen wird; und
der erste Knoten eingerichtet ist, es zu verhindern, dass die erste Kopie der Ressource verändert wird, nachdem die zweite Kopie zu dem zweiten Zwischenspeicher übertragen worden ist.

26. System gemäß Anspruch 23, wobei
der erste Knoten eingerichtet ist, eine Anforderung der Genehmigung zu senden für die erste Kopie nach dem Übertragen der zweiten Kopie zu dem zweiten Zwischenspeicher an den Hauptknoten (D) freizugeben; und
der erste Knoten eingerichtet ist, eine Nachricht von dem Hauptknoten (D) zu empfangen, die anzeigt, dass die erste Kopie freigegeben werden kann, nachdem der Hauptknoten (D) es als Reaktion auf die Anforderung veranlasst, dass die erste Kopie oder ein Versionsnachfolger von ihr dauerhaft gespeichert wird.

27. System gemäß Anspruch 26, wobei
der erste Knoten einen Sendeprozess aufweist, der eingerichtet ist, eine Anforderung für die Genehmigung für das Freigeben der ersten Kopie zu senden; und
ein anderer Prozess als der Sendeprozess einen Versionsnachfolger der ersten Kopie der Ressource speichert.

28. System gemäß Anspruch 23, wobei der erste Knoten eingerichtet ist, mindestens eine Kopie der Ressource in dem ersten Zwischenspeicher beizubehalten, durch:
Bestimmen, ob eine dauerhaft gespeicherte Kopie der Ressource neuer ist als die erste Kopie, bevor versucht wird, die erste Kopie dauerhaft zu speichern;
Freigeben der ersten Kopie ohne dauerhaftes Speichern der ersten Kopie, falls die dauerhaft gespeicherte Kopie neuer als die erste Kopie ist; und
dauerhaftes Speichern der ersten Kopie, falls die dauerhaft gespeicherte Kopie nicht neuer als die erste Kopie ist.

29. System gemäß Anspruch 25, wobei der erste Knoten eingerichtet ist, eine Änderungsgenehmigung von einem Sendeprozess, der dem ersten Zwischenspeicher zugeordnet ist (204), zu einem Empfangsprozess, der dem zweiten Zwischenspeicher zugeordnet ist, zusammen mit der zweiten Kopie der Ressource zu übertragen.

30. System gemäß Anspruch 29, wobei:
Genehmigungen für das Zugreifen auf die Ressource mittels eines Hauptservers (D) verwaltet werden; und
der erste Knoten eingerichtet ist, die Änderungsgenehmigung zu dem Empfangsprozess zu übertragen, bevor der erste Knoten die Bestätigung von dem Hauptserver (D) für die Übertragung der Änderungsgenehmigung zu dem Empfangsprozess empfängt.

31. System gemäß Anspruch 23, wobei:
der erste Knoten eingerichtet ist, eine Änderungsgenehmigung von einem Sendeprozess, der dem ersten Zwischenspeicher zugeordnet ist, an einen Empfangsprozess, der dem zweiten Zwischenspeicher zugeordnet ist, zusammen mit der zweiten Kopie der Ressource zu übertragen;
Genehmigungen für das Zugreifen auf die Ressource mittels eines Hauptservers (D) verwaltet werden; und
das Übertragen der Änderungsgenehmigung zu dem Empfangsprozess durchgeführt wird, bevor der Hauptserver (D) die Bestätigung für die Übertragung der Änderungsgenehmigung zu dem Empfangsprozess empfängt.

32. System gemäß Anspruch 23, wobei:
der erste Knoten einen Sendeprozess aufweist, der

dem ersten Zwischenspeicher zugeordnet ist, wobei der Sendeprozess eingerichtet ist, eine Nachricht von einem Hauptknoten (D) zu empfangen, der eine Anforderung für die Ressource von einem Empfangsprozess, der dem zweiten Zwischenspeicher zugeordnet ist, empfängt; und
der Sendeprozess die zweite Kopie zu dem Empfangsprozess als Reaktion auf die Nachricht von dem Hauptserver (D) überträgt.

33. Verfahren gemäß Anspruch 23, wobei:
der erste Knoten einen Sendeprozess aufweist, der eingerichtet ist, eine Sperre von einem Sperrmanager anzufordern, nachdem die zweite Kopie zu dem zweiten Zwischenspeicher übertragen ist, wobei die Sperre die Genehmigung zum Schreiben der Ressource auf Platte (**655**) erteilt, aber nicht die Genehmigung zur Änderung der Ressource; und
der Sperrmanager einen Prozess auswählt, der eine Version der Ressource aufweist, die mindestens so neu wie die erste Kopie ist und dem ausgewählten Prozess die Sperre erteilt, um den ausgewählten Prozess zu veranlassen, die Version der Ressource auf Platte (**655**) zu schreiben.

34. System gemäß Anspruch 23, wobei der Sperrmanager als Reaktion auf das Schreiben der Version der Ressource auf Platte (**655**) veranlasst, dass alle Versionen der Ressource, die älter als die Version sind, freigegeben werden.

35. System gemäß Anspruch 23, ferner aufweisend:
einen Hauptknoten (D), der eingerichtet ist zu bestimmen, nach einem Ausfall eines Zwischenspeichers, der eine verschmutzte Kopie der Ressource enthält, ob ein ausgefallener Zwischenspeicher die letzte Version der Ressource enthalten hat; und
wobei, falls der ausgefallene Zwischenspeicher die letzte Version der Ressource enthalten hat, der Hauptknoten (D) eingerichtet ist zu veranlassen, dass ein letztes früheres Abbildbild der Ressource auf Platte (**655**) geschrieben wird, zu veranlassen, dass alle vorherigen früheren Abbilder der Ressource freigegeben werden, und zu veranlassen, dass ein Wiederherstellungsprotokoll des ausgefallenen Zwischenspeichers verwendet wird, um die letzte Version der Ressource zu rekonstruieren.

36. System gemäß Anspruch 35, wobei der Hauptknoten (D) eingerichtet ist, falls der ausgefallene Zwischenspeicher nicht die letzte Version der Ressource enthalten hat, zu veranlassen, dass die letzte Version der Ressource auf Platte (**655**) geschrieben wird und zu veranlassen, dass alle früheren Abbilder der Ressource freigegeben werden.

37. Verfahren gemäß Anspruch 23, ferner aufweisend: einen Hauptknoten (D), der eingerichtet ist, nach einem Ausfall einer Mehrzahl von Zwischenspeichern, die verschmutzte Versionen der Ressource enthalten, zu bestimmen, ob irgendeiner der ausgefallenen Zwischenspeicher die letzte Version der Ressource enthalten hat und, falls irgendeiner der ausgefallenen Zwischenspeicher die letzte Version der Ressource enthalten hat, die Wiederherstellungsprotokolle der ausgefallenen Zwischenspeicher zusammenzuführen und zu verwenden, um die letzte Version der Ressource zu rekonstruieren.

38. System gemäß Anspruch 23, wobei der erste Zwischenspeicher und der zweite Zwischenspeicher zu einer Mehrzahl von Zwischenspeichern gehören, wobei ein separates Wiederherstellungsprotokoll für jeden Zwischenspeicher der Mehrzahl von Zwischenspeichern verwaltet wird und das System ferner aufweist:
einen Hauptknoten (D), der eingerichtet ist, einen ausgefallenen Zwischenspeicher der Mehrzahl von Zwischenspeichern wiederherzustellen, basierend auf dem Wiederherstellungsprotokoll, das dem ausgefallenen Zwischenspeicher zugeordnet ist, ohne die separaten Wiederherstellungsprotokolle der anderen Zwischenspeicher der Mehrzahl von Zwischenspeichern zu überprüfen.

39. System gemäß Anspruch 38, wobei jeder Zwischenspeicher der Mehrzahl von Zwischenspeichern ein Zwischenspeicher ist, der von einem separaten Datenbank-Server einer Mehrzahl von Datenbank-Servern (A, B, C, D) verwaltet wird.

40. System gemäß Anspruch 38, wobei
der erste Knoten eingerichtet ist, es zu ermöglichen, dass die erste Kopie der Ressource in dem ersten Zwischenspeicher verändert wird, bevor die zweite Kopie zu dem zweiten Zwischenspeicher übertragen wird; und
der erste Knoten eingerichtet ist zu verhindern, dass die erste Kopie der Ressource verändert wird, nachdem die zweite Kopie zu dem zweiten Zwischenspeicher übertragen worden ist.

41. Verfahren gemäß Anspruch 23, wobei
der erste Knoten eingerichtet ist:
die Ressource in dem ersten Zwischenspeicher des ersten Knotens zu modifizieren, so dass eine geänderte Version der Ressource erzeugt wird;
einen Punktpunkt für den ersten Knoten zu verwalten, der anzeigt, wo mit der Arbeit begonnen werden muss, wenn der erste Knoten ausfällt;
eine ersten Kopie der geänderten Version in dem ersten Zwischenspeicher beizubehalten während des Übertragens einer zweiten Kopie der geänderten Version von dem ersten Zwischenspeicher zu dem zweiten Zwischenspeicher des zweiten Knotens, ohne die geänderte Version von dem ersten Zwischenspeicher vorher dauerhaft in einen persistenten Speicher zu speichern; und
den Prüfpunkt als Reaktion auf ein Hinweis, dass ein

anderer Knoten der Mehrzahl von Knoten (A, B, C, D) eine Version der Ressource dauerhaft gespeichert hat, die mindestens so neu wie die geänderte Version ist, vorzuversetzen.

42. System gemäß Anspruch 41, wobei der erste Knoten ferner eingerichtet ist:
mindestens eine Kopie der geänderten Version in dem ersten Zwischenspeicher beizubehalten, bis die erste Kopie der geänderten Version oder ein Versionsnachfolger von ihr dauerhaft gespeichert ist.

43. System gemäß Anspruch 41, wobei der erste Knoten ferner eingerichtet ist:
die Ressource an dem ersten Knoten freizugeben als Reaktion auf den Hinweis, dass ein anderer Knoten der Mehrzahl von Knoten (A, B, C, D) die Version der Ressource dauerhaft gespeichert hat, die mindestens so neu wie die geänderte Version ist.

44. System gemäß Anspruch 23,
wobei der erste Knoten eingerichtet ist zu verhindern, dass die erste Kopie in dem ersten Zwischenspeicher ersetzt wird, bis die erste Kopie oder ein Versionsnachfolger von ihr dauerhaft gespeichert ist.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

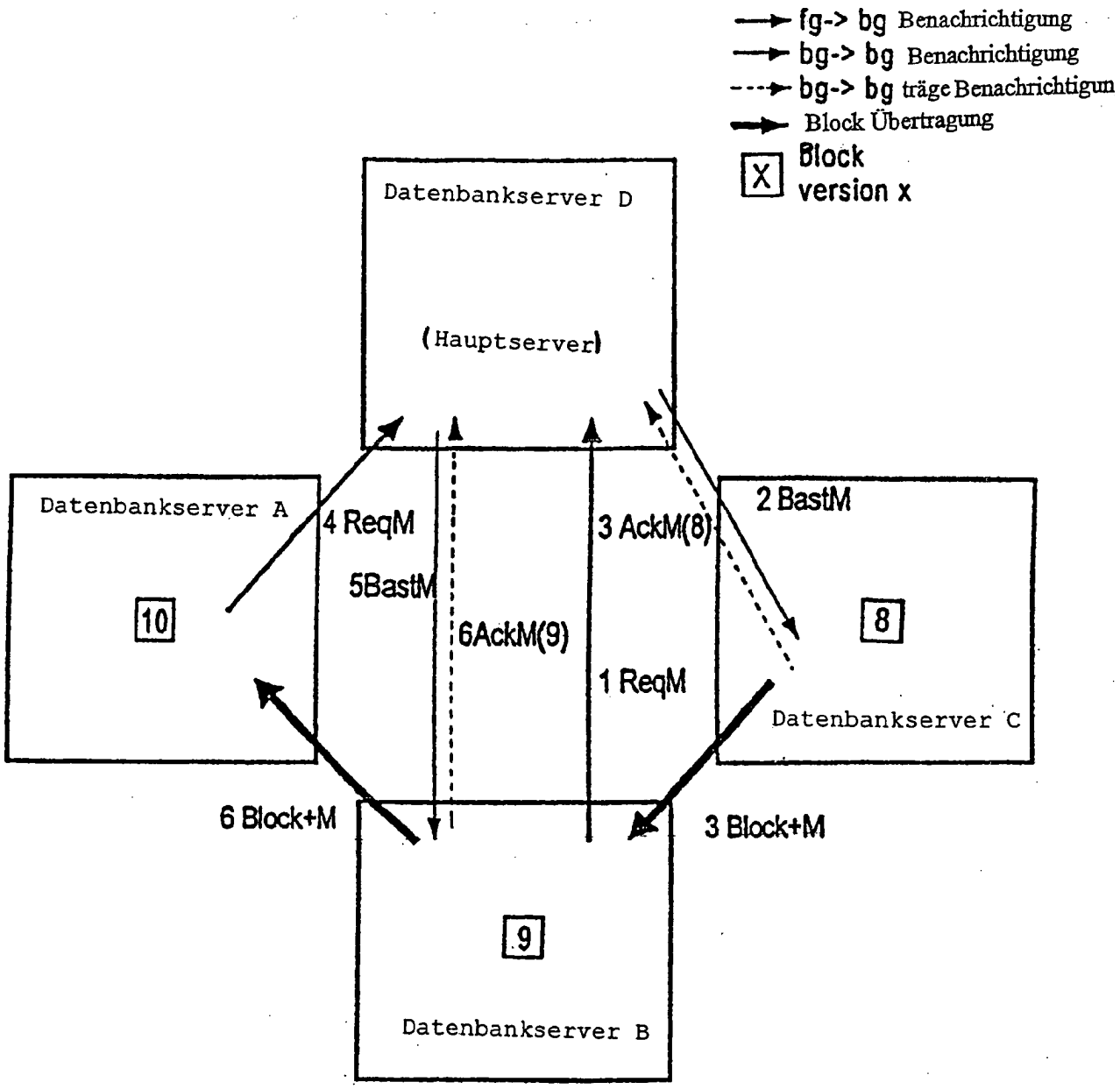


Fig. 1

Fig. 2

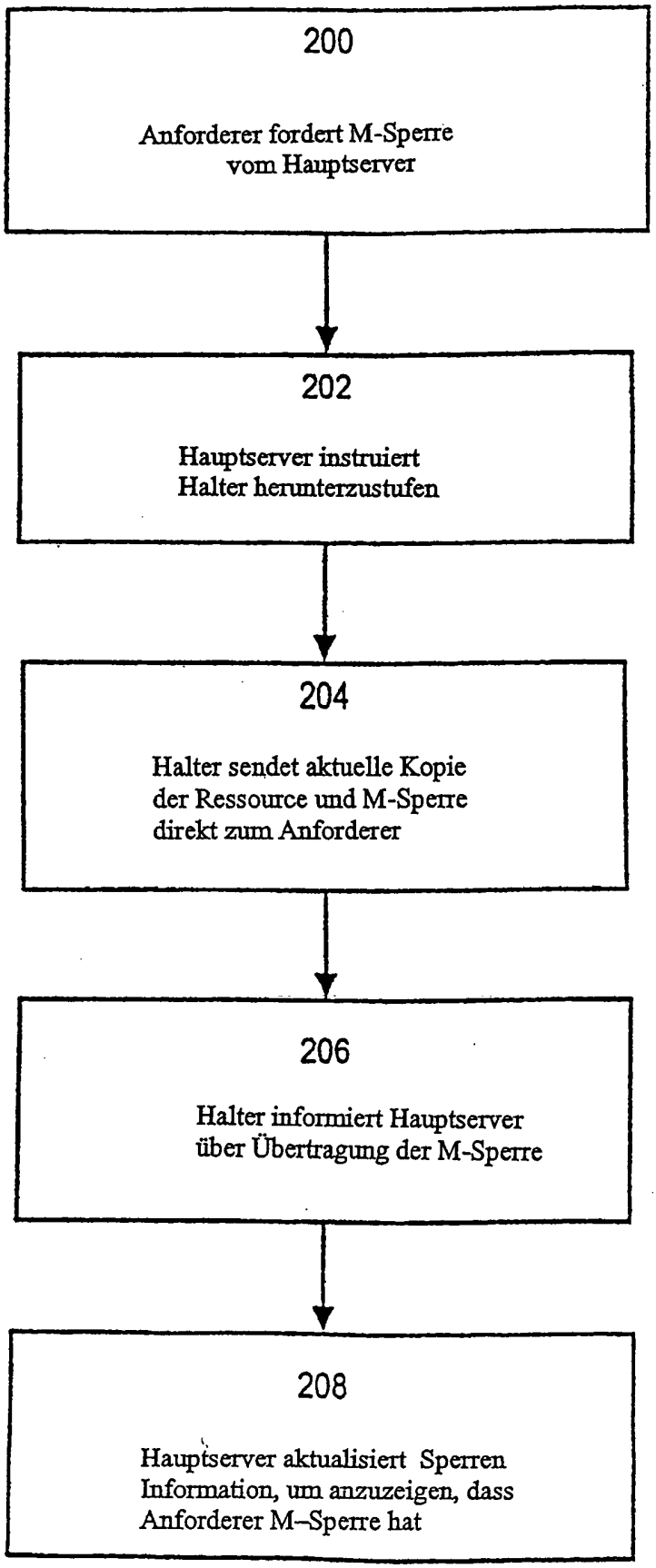

200
Anforderer fordert M-Sperre vom Hauptserver
202
Hauptserver instruiert Halter herunterzustufen
204
Halter sendet aktuelle Kopie der Ressource und M-Sperre direkt zum Anforderer
206
Halter informiert Hauptserver über Übertragung der M-Sperre
208
Hauptserver aktualisiert Sperren Information, um anzuzeigen, dass Anforderer M–Sperre hat

Fig. 3

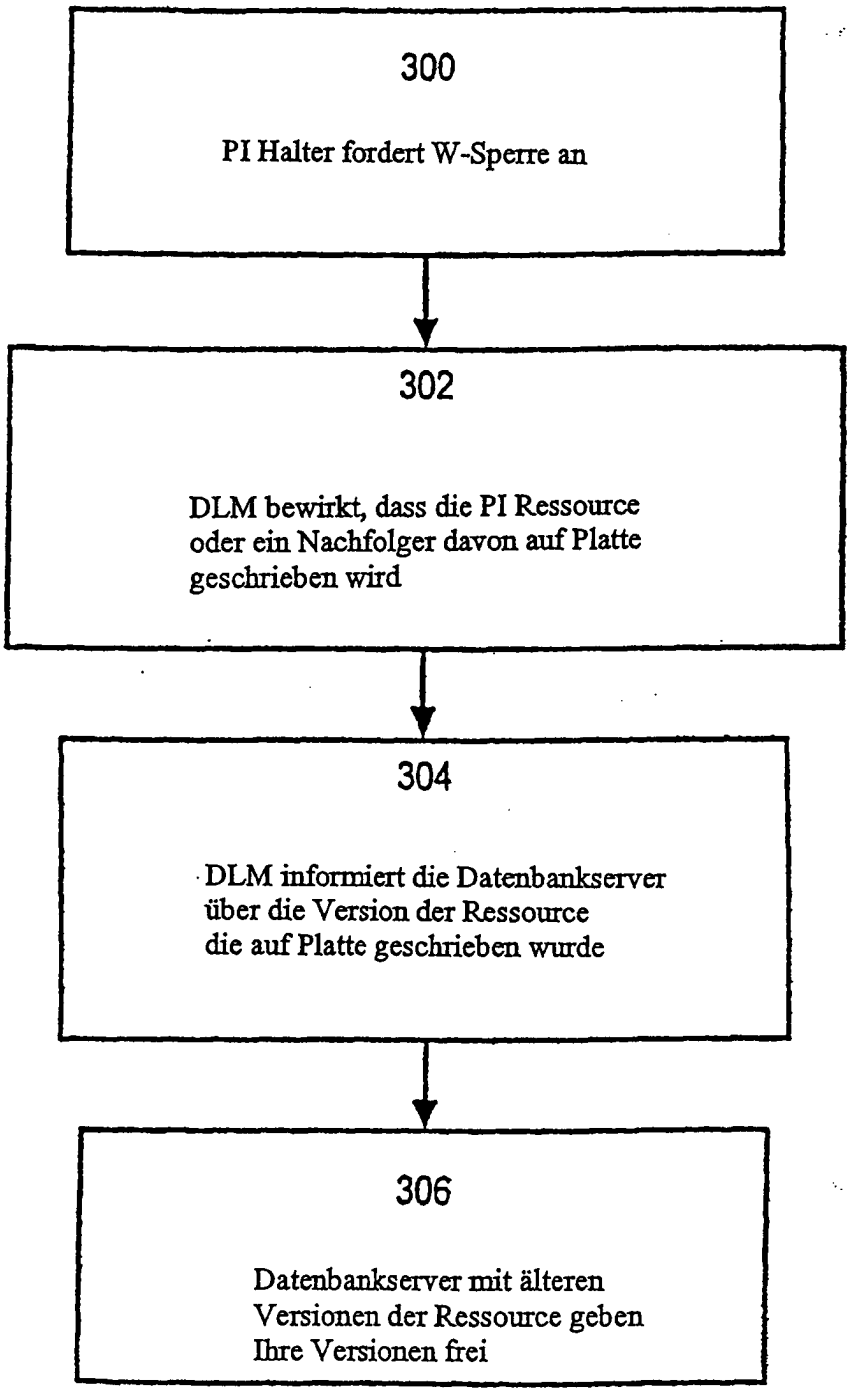
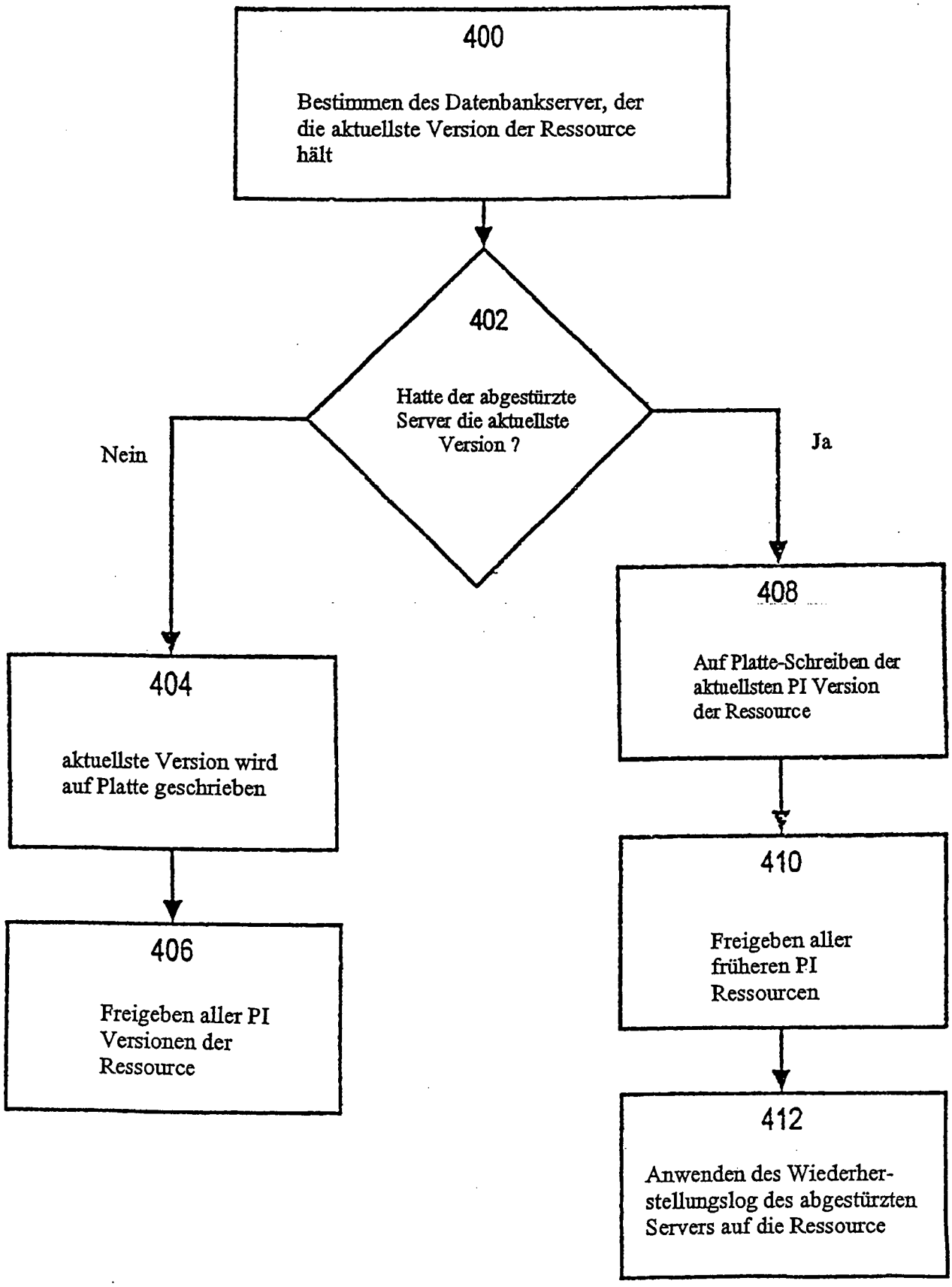

300
PI Halter fordert W-Sperre an
302
DLM bewirkt, dass die PI Ressource oder ein Nachfolger davon auf Platte geschrieben wird
304
DLM informiert die Datenbankserver über die Version der Ressource die auf Platte geschrieben wurde
306
Datenbankserver mit älteren Versionen der Ressource geben Ihre Versionen frei

Fig. 4


400
Bestimmen des Datenbankserver, der die aktuellste Version der Ressource hält
402
Hatte der abgestürzte Server die aktuellste Version ?
Nein
Ja
404
aktuellste Version wird auf Platte geschrieben
406
Freigeben aller PI Versionen der Ressource
408
Auf Platte-Schreiben der aktuellsten PI Version der Ressource
410
Freigeben aller früheren PI Ressourcen
412
Anwenden des Wiederherstellungslog des abgestürzten Servers auf die Ressource

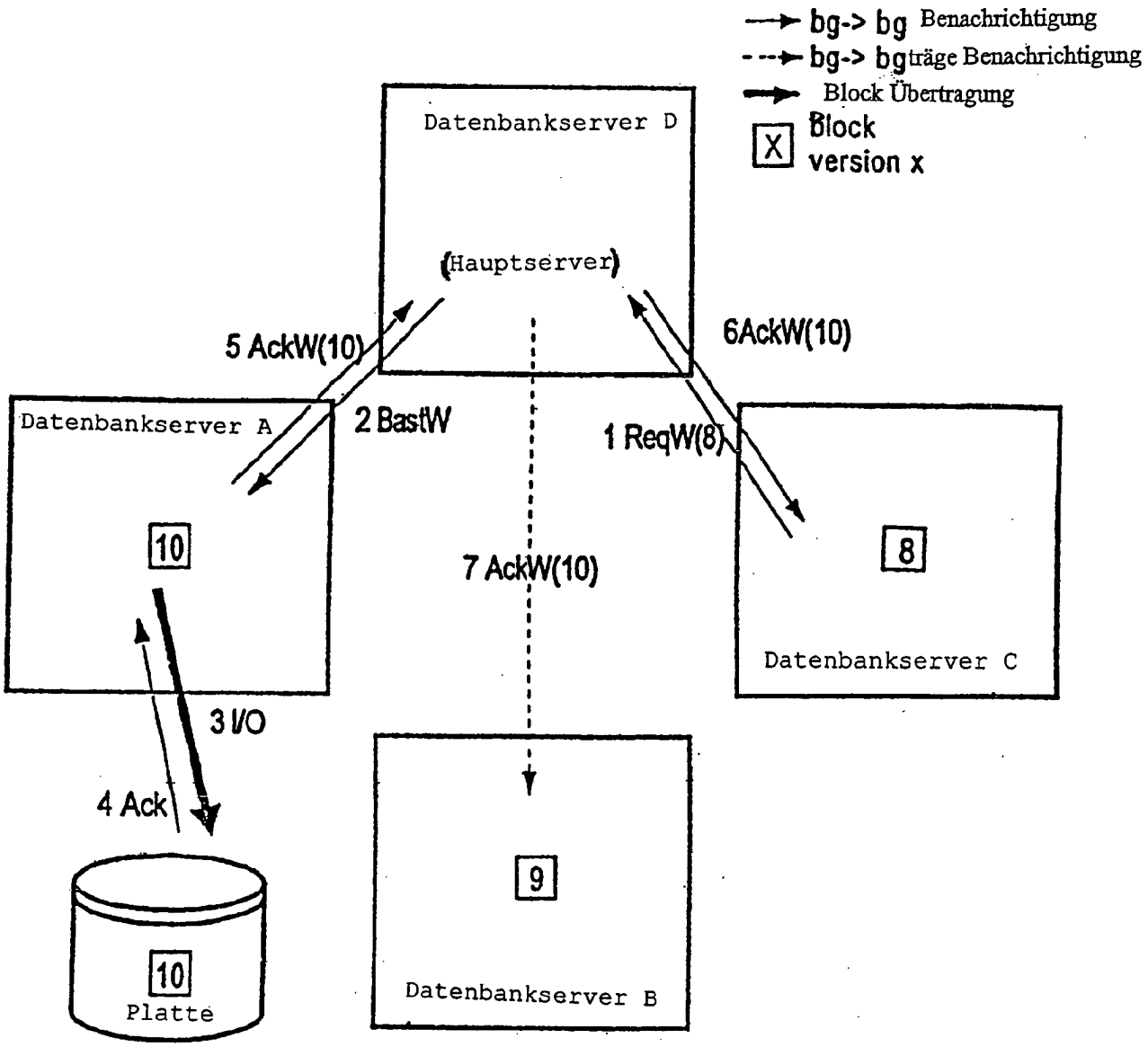

bg-> bg Benachrichtigung
bg-> bg träge Benachrichtigung
Block Übertragung
X Block version x
Datenbankserver D
(Hauptserver)
5 AckW(10)
2 BastW
6AckW(10)
1 ReqW(8)
Datenbankserver A
10
7 AckW(10)
8
Datenbankserver C
3 I/O
4 Ack
9
10
Platte
Datenbankserver B

Fig. 5

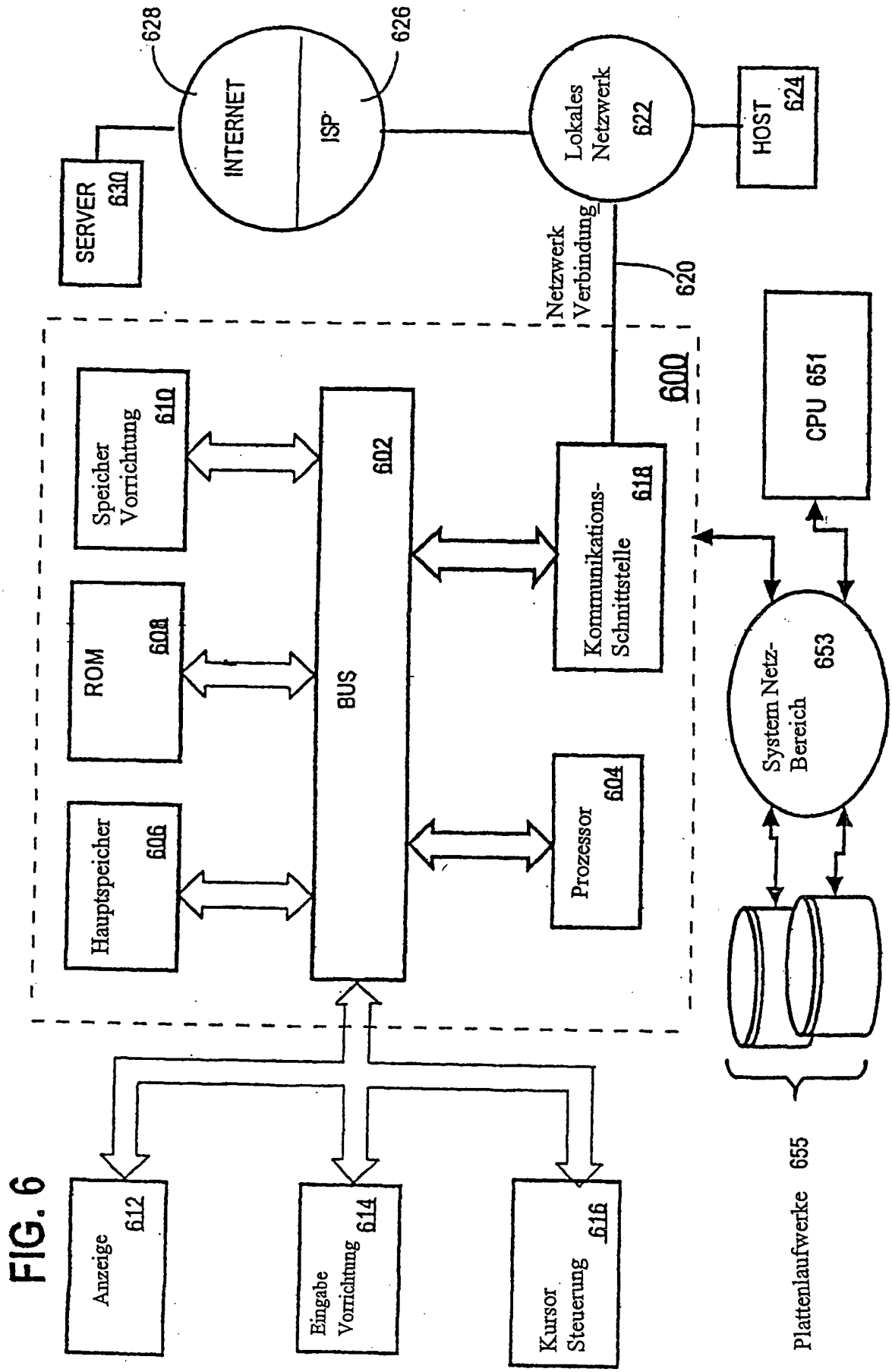

628
INTERNET
ISP
626
SERVER
630
Lokales Netzwerk
622
HOST
624
Netzwerk Verbindung
620
600
CPU 651
Speicher Vorrichtung
610
602
Kommunikations-Schnittstelle
618
ROM
608
BUS
System Netz-Bereich
653
Hauptspeicher
606
Prozessor
604
Anzeige
612
Eingabe Vorrichtung
614
Kursor Steuerung
616
Plattenlaufwerke
655

FIG. 6