(19) Deutsches Patent- und Markenamt

(10) **DE 10 2015 007 709 B4** 2025.05.22

(12) # Patentschrift

(21) Aktenzeichen: **10 2015 007 709.0**
(22) Anmeldetag: **17.06.2015**
(43) Offenlegungstag: **31.12.2015**
(45) Veröffentlichungstag der Patenterteilung: **22.05.2025**

(51) Int Cl.: ***G06F 12/12*** (2006.01)
G06F 11/07 (2006.01)

Innerhalb von neun Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten (§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 1 Patentkostengesetz).

(30) Unionspriorität:
14/316,256 **26.06.2014** **US**

(73) Patentinhaber:
Western Digital Technologies, Inc., San Jose, CA, US

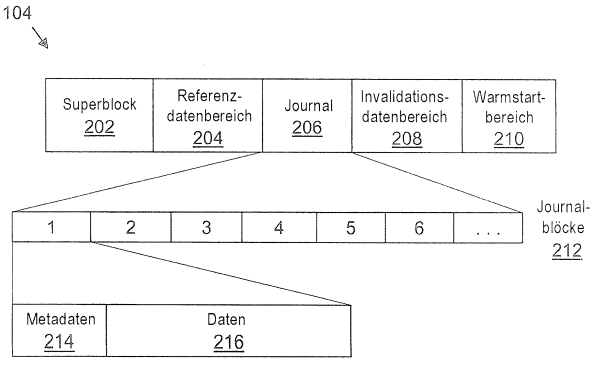
(74) Vertreter:
Murgitroyd Germany Patentanwaltsgesellschaft mbH, 80636 München, DE

(72) Erfinder:
Misra, Pulkit, San Jose, Calif., US

(56) Ermittelter Stand der Technik:
US **2014 / 0 047 193** **A1**

(54) Bezeichnung: **Invalidationsdatenbereich für einen Cache**

(57) Hauptanspruch: Cache (104), der umfasst:
ein Journal (206), das zum Nachverfolgen von in dem Cache (104) gespeicherten Datenblöcken (216) ausgelegt ist; und
einen Invalidationsdatenbereich (208), der zum Nachverfolgen von invalidierten Datenblöcken ausgelegt ist, die den Datenblöcken (216) zugeordnet sind, welche in dem Journal (206) nachverfolgt werden, wobei sich der Invalidationsdatenbereich (208) in einer von dem Journal (206) getrennten Region des Caches (104) befindet.



Beschreibung

HINTERGRUND

Gebiet der Offenlegung

[0001] Die vorliegende Offenlegung betrifft Systeme und Verfahren zum Cachen und insbesondere das Bereitstellen einer Region zum Verarbeiten von invalidierten Daten für einen Cache.

Verwandte Offenlegung

[0002] Ein Cache kann generell verwendet werden, um einen Zugriff beim Lesen oder Schreiben von Daten in eine zugrundeliegende Speicherungseinrichtung, wie z. B. einen Flashspeicher oder eine Festplatte, zu beschleunigen. Bei Empfang einer Schreiboperation aus einem Host kann der Cache einen gespeicherten Datenblock aktualisieren, um nachzuverfolgen, ob sich der Datenblock verändert hat (z. B. ob der Datenblock valid oder nichtvalid ist). Manchmal kann der Cache die neuen Daten aus der Schreiboperation in einen anderen Cache-Eintrag schreiben und das Räumen oder Löschen des alten Cache-Eintrags aufschieben. Der Grund dafür ist, dass das Räumen oder Löschen des alten Cache-Eintrags eine Abnahme der Leistung bewirken kann, während der Cache darauf wartet, dass die zugrundeliegende Speicherungseinrichtung aktualisiert wird. Unter Ausnutzung dieser Aufschiebung kann der Cache das Verarbeiten der Schreiboperation beenden und eine schnellere Steuerungsrückführung zu dem Host durchführen.

[0003] Weiterhin kann die Offenbarung der US 2014 / 0 047 193 A1 gegebenenfalls für das Verständnis der vorliegenden Erfindung hilfreich sein. In dieser Druckschrift wird ein Computersystem mit einem Cache mit einem oder mehreren Speichern, einem Cache-Journal zum Speichern von Daten, die einem oder mehreren Teilen des Caches zugeordnet sind, und einem Konfigurationsmanager für den Zugriff auf den Cache und das Cache-Journal beschrieben. Der Konfigurationsmanager kann feststellen, ob das Cache-Journal Daten enthält, die einem ersten Teil des Caches zugeordnet sind, und im Cache-Journal Daten erzeugen, die dem ersten Teil des Caches zugeordnet sind, wenn das Cache-Journal noch keine Daten enthält, die dem ersten Teil des Caches zugeordnet sind. Der Konfigurationsmanager ist auch in der Lage zu bestimmen, ob der erste Teil des Caches für die Verwendung gültig ist, und mit einem Speichermanager, der mit dem ersten Teil des Caches verbunden ist, darüber zu kommunizieren, ob der erste Teil des Caches für die Verwendung gültig ist.

KURZFASSUNG

[0004] Ausführungsformen der vorliegenden Offenlegung betreffen Caches, Verfahren und Systeme zum Verwenden eines Invalidationsdatenbereichs.

[0005] Insbesondere betrifft die vorliegende Erfindung einen Cache gemäß Anspruch 1. Vorteilhafte Ausführungsformen können Merkmale abhängiger Ansprüche aufweisen.

[0006] Bei einer Ausführungsform betrifft die vorliegende Erfindung demzufolge einen Cache. der ein Journal und einen Invalidationsdatenbereich aufweist. Das Journal ist zum Nachverfolgen von Datenblöcken, die in dem Cache gespeichert sind, ausgelegt. Der Invalidationsdatenbereich ist zum Nachverfolgen von invalidierten Datenblöcken, die den in dem Journal nachverfolgten Datenblöcken zugeordnet sind, ausgelegt, wobei sich der Invalidationsdatenbereich in einer von dem Journal getrennten Region des Caches befindet.

[0007] Die hier beschriebenen Ausführungsformen können weitere Aspekte umfassen. Zum Beispiel kann das Journal zum Nachverfolgen von Metadaten für die Datenblöcke ausgelegt sein, wobei die Metadaten eine Speicheradresse aufweisen können, die dem Datenblock entspricht, und kann der Invalidationsdatenbereich so ausgelegt sein, dass er Metadaten nachverfolgt, die den invalidierten Datenblöcken entsprechen, wobei die zugeordneten Metadaten eine Speicheradresse aufweisen können, die dem invalidierten Datenblock entspricht. Das Journal kann zum Nachverfolgen der Datenblöcke unter Verwendung von Journalblöcken ausgelegt sein, wobei die Journalblöcke zum Speichern der Metadaten für die Datenblöcke ausgelegt sein können, und der Invalidationsdatenbereich zum Nachverfolgen der Metadaten, die den nichtvaliden Datenblöcken zugeordnet sind, unter Verwendung von Invalidationsaufzeichnungen und abgebildeten Journalblöcken ausgelegt sein kann, wobei die abgebildeten Journalblöcke zum Speichern der zugeordneten Metadaten für die nichtvaliden Datenblöcke ausgelegt sein können und wobei die Invalidationsaufzeichnungen zum Speichern der abgebildeten Journalblöcke ausgelegt sein können. Die in dem Journal nachverfolg-

ten Metadaten können ferner einen Index in eine Ansammlung von Metadaten, die in jedem Journalblock gespeichert sind, aufweisen, und die in dem Invalidationsdatenbereich nachverfolgten Metadaten können ferner einen Index in eine Ansammlung von Metadaten, die in jedem abgebildeten Journalblock gespeichert sind, aufweisen. Der Cache kann so ausgelegt sein, dass er eine Invalidationsaufzeichnungsnummer, die einer Invalidationsaufzeichnung in dem Invalidationsdatenbereich zugeordnet ist, auf der Basis einer entsprechenden Journalblocknummer, die einem Journalblock zugeordnet ist, ermittelt. Der Cache kann so ausgelegt sein, dass er eine Abgebildet-Journalblock-Nummer, die einem abgebildeten Journalblock in dem Invalidationsdatenbereich zugeordnet ist, auf der Basis einer entsprechenden Journalblocknummer, die einem Journalblock zugeordnet ist, ermittelt. Der in dem Journal nachverfolgte Index kann so ausgewählt sein, dass er den gleichen Wert aufweist wie der Index, der in dem Invalidationsdatenbereich nachverfolgt wird. Die in dem Invalidationsdatenbereich nachverfolgte Speicheradresse kann im Vergleich zu der in dem Journal nachverfolgten Speicheradresse verkürzt sein, und die Verkürzung kann ermittelt werden auf der Basis einer Speicherungsgröße einer zugrundeliegenden Speicherungsvorrichtung, die gecacht wird, oder eines Offsets, der auf der Basis einer Speicheradresse eines Blocks in der zugrundeliegenden Speicherungsvorrichtung ermittelt wird. Das Ermitteln des abgebildeten Journalblocks kann umfassen: Ermitteln einer Abgebildet-Journalblock-Nummer für den abgebildeten Journalblock durch Ermitteln einer Invalidationsaufzeichnungsnummer durch Dividieren einer Journalblocknummer, die dem ermittelten Journalblock zugeordnet ist, durch eine Kapazität der Invalidationsaufzeichnung in dem Invalidationsdatenbereich und Berechnen einer Rundungsfunktion des Ergebnisses der Division, wobei die Invalidationsaufzeichnungsnummer die Invalidationsaufzeichnung identifiziert, und Ermitteln der Abgebildet-Journalblock-Nummer durch Berechnen einer Modulo-Operation der Journalblocknummer mit der Kapazität der Invalidationsaufzeichnung. Das Ermitteln, ob Schreiboperationen ausstehen, kann das Aufrufen eines Felds aus einer in-RAM-Datenstruktur, die der Invalidationsaufzeichnung entspricht, umfassen. Das Vereinigen der ausstehenden Schreiboperationen kann umfassen: Einreihen nachfolgender Schreiboperationen in eine Warteschlange, Identifizieren von Schreiboperationen, die auf demselben Datenblock arbeiten, und Ermitteln der einzelnen Schreiboperation auf der Basis der Schreiboperationen, die auf demselben Datenblock arbeiten. Der Invalidationsdatenbereich kann sich in einer von dem Journal getrennten Region des Caches befinden. Der Cache kann ein Inhaltliche-Lokalität-Cache sein, und das Journal kann mindestens einen der zugeordneten Datenblöcke und unabhängigen Datenblöcke in dem Inhaltliche-Lokalität-Cache nachverfolgen. Das Ermitteln der Anfangs-Rekonstruktion kann das Wiederherstellen von Datenblöcken und Metadaten, die die Datenblöcke beschreiben, umfassen, wobei die wiederhergestellten Datenblöcke und Metadaten aus dem Journal wiederhegestellt werden. Das Ermitteln, ob der entsprechende Datenblock, der in dem Journal nachverfolgt wird, valid ist, kann das Vergleichen von Metadaten, die den entsprechenden in dem Journal nachverfolgten Datenblock beschreiben, mit Metadaten, die den entsprechenden in dem abgebildeten Journalblock nachverfolgten Datenblock beschreiben, umfassen. Das Vergleichen der Metadaten kann das Vergleichen einer ersten Speicheradresse und eines ersten Indexes für den entsprechenden Datenblock, der in dem Journal nachverfolgt wird, mit einem zweiten Speicher und einem zweiten Index, die in dem abgebildeten Journalblock nachverfolgt werden, umfassen.

KURZBESCHREIBUNG DER FIGUREN

[0008] Verschiedene Aufgaben, Merkmale und Vorteile der vorliegenden Offenlegung werden mit Bezug auf die folgende detaillierte Beschreibung in Zusammenhang mit den folgenden Zeichnungen besser verständlich, in denen gleiche Bezugszeichen gleiche Elemente identifizieren. Die folgenden Zeichnungen dienen nur zum Zweck der Veranschaulichung und dürfen nicht als die Erfindung einschränkend verstanden werden, deren Umfang in den nachfolgenden Patentansprüchen dargelegt ist.

Fig. 1 zeigt ein beispielhaftes System mit einem Cache gemäß einigen Ausführungsformen der vorliegenden Offenlegung.

Fig. 2A-2B zeigen beispielhafte Blockschaltbilder eines Caches gemäß einigen Ausführungsformen der vorliegenden Offenlegung.

Fig. 3A-3B zeigen beispielhafte Abbildungen zwischen einem Journal und einem Invalidationsdatenbereich gemäß einigen Ausführungsformen der vorliegenden Offenlegung.

Fig. 4 zeigt ein beispielhaftes Verfahren zum Invalidieren unter Verwendung eines Invalidationsdatenbereichs gemäß einigen Ausführungsformen der vorliegenden Offenlegung.

Fig. 5 zeigt ein beispielhaftes Verfahren für eine Cache-Wiederherstellung gemäß einigen Ausführungsformen der vorliegenden Offenlegung.

DETAILLIERTE BESCHREIBUNG

[0009] Die vorliegende Offenlegung betrifft Systeme und Verfahren zum Verwenden eines Invalidationsdatenbereichs für einen Cache. Bei einigen Ausführungsformen kann der Cache einen Journalbereich und einen Invalidationsdatenbereich aufweisen. Der Journalbereich kann ein protokollbasiertes Journal zum beständigen Nachverfolgen von Cache-Aktualisierungen und Cache-Operationen im Fall eines Erfordernisses einer Cache-Wiederherstellung sein. In dem Invalidationsdatenbereich können Invalidationsaufzeichnungen für Cache-Blöcke gespeichert sein, die aus dem Cache entfernt oder geräumt werden. Der Invalidationsdatenbereich kann generell Informationen über gecachte Datenblöcke nachverfolgen, die invalidiert worden sind, zum Beispiel während das Cachen pausiert oder anderweitig unterbrochen ist. Der Invalidationsdatenbereich kann dem Journalbereich angefügt sein und eine separate Region des Caches einnehmen. Ferner kann bei einigen Ausführungsformen des Invalidationsdatenbereichs einen Teilsatz von Metadaten, die einem generell in dem Journal gespeicherten vollen Satz von Metadaten entsprechen, gespeichert werden.

[0010] Fig. 1 zeigt ein beispielhaftes System 100, das einen Cache 104 aufweist, gemäß einigen Ausführungsformen der vorliegenden Offenlegung. Das System 100 weist einen Host 102, den Cache 104 und eine Speicherungseinrichtung 106a-106c auf. Der Host 102 überträgt Lese- und Schreibanforderungen an den Cache 104. Der Cache 104 verarbeitet die Anforderungen zum Lesen und Schreiben von Daten in die und aus der zugrundeliegenden Speicherungseinrichtung 106a-106c. Zum Beispiel kann zum Verarbeiten einer Leseanforderung der Cache 104 ermitteln, ob Daten, die einer angeforderten Speicheradresse entsprechen, in dem Cache gespeichert sind. Falls die angeforderte Speicheradresse gecacht ist, kann diese Situation manchmal als „Lese-Treffer“ bezeichnet werden. Falls die angeforderte Speicheradresse nicht gecacht ist, kann diese Situation als „Lese-Nichttreffer“ bezeichnet werden. Bei einem Lese-Treffer kann der Cache 104 die angeforderten Daten schneller direkt aus dem Cache 104 zurückführen. Im Gegensatz dazu kann bei einem „Lese-Nichttreffer“ der Cache 104 die angeforderten Daten aus der langsameren Speicherungseinrichtung 106a-106c lesen.

[0011] Auf im Wesentlichen gleiche Weise kann zum Verarbeiten einer Schreibanforderung der Cache 104 ermitteln, ob eine angeforderte Speicheradresse bereits in dem Cache gespeichert ist. Falls die angeforderte Speicheradresse gecacht ist, kann diese Situation manchmal als „Schreib-Treffer“ bezeichnet werden. Falls die angeforderte Speicheradresse nicht gecacht ist, kann diese Situation als „Schreib-Nichttreffer“ bezeichnet werden.

[0012] Fig. 2A zeigt ein beispielhaftes Blockschaltbild des Caches 104 gemäß einigen Ausführungsformen der vorliegenden Offenlegung. Bei einigen Ausführungsformen kann der Cache 104 einen Superblock 202, einen Referenz-Datenbereich 204, ein Journal 206, einen Invalidationsdatenbereich 208 und einen Warmstartbereich 210 aufweisen. Das Journal 206 kann Journalblöcke 212 aufweisen. Die Journalblöcke 212 können Metadaten 214 und Daten 216 aufweisen.

[0013] Der Cache 104 kann eine journalbasierte Vorgehensweise anwenden, um eine Beständigkeit zu bieten, so dass der Cache 104 erforderlichenfalls wiederhergestellt werden kann. Einige Ausführungsformen des Journals 206 können in Journalblöcke 212 unterteilt sein. Zum Beispiel können die Journalblöcke 212 eine Größe von ungefähr 256 kB aufweisen. Andere Größen relativ zu der Gesamtgröße des Caches 104 können ebenfalls verwendet werden. Falls ein Journalblock 212 eine Größe von ungefähr 256 kB aufweist, können die Metadaten 214 eine Größe von ungefähr 4 kB einnehmen und können die Daten 216 ungefähr 252 kB nutzen. Wie zuvor gesagt, können je nach Bedarf des Journals 206 und des Caches 104 auch andere Größen verwendet werden.

[0014] Die Daten 216 können den Inhalt aufweisen, der einem Cache-Block zugeordnet ist, welcher in dem Journal 206 nachverfolgt wird. Beispiele für die Metadaten 214 können eine Speicheradresse (z. B. eine Logikblockadresse (LBA)), einen Cache-Block-Typ, einen Offset und einen Hash-Wert für eine Fehlerkorrektur umfassen. Ein Beispiel für einen Cache-Block-Typ kann das Nachverfolgen umfassen, dass ein Cache-Block ein unabhängiger Block oder ein zugeordneter Block ist. Ein unabhängiger Block und/oder ein zugeordneter Block können bei einem Inhaltliche-Lokalität-Cache verwendet werden. Bei einigen Ausführungsformen kann der Cache 104 auf der Basis einer Ähnlichkeit eines Cache-Blocks (inhaltliche Lokalität) ein Cachen durchführen. Ein zugeordneter Block kann Veränderungen oder Deltas zwischen Baseline-Referenzblöcken nachverfolgen. Dieses Inhaltliche-Lokalität-Cachen kann zusätzlich zum Ermitteln, wann ein Cache-Block zuletzt verwendet worden ist (zeitliche Lokalität), oder zum Identifizieren von Cache-Blöcken mit im Wesentlichen gleichen Speicheradressen (räumliche Lokalität) erfolgen. Ein unabhängiger Block kann ein Block sein, der auf der Basis einer zeitlichen Lokalität und/oder einer räumlichen Lokalität, jedoch keiner inhaltlichen

Lokalität gecacht wird. Der Offset kann einen spezifischen interessierenden Speicherblock oder eine spezifische interessierende Speicherstelle in einem Speicherblock identifizieren. Zum Beispiel kann der Offset einem Zeiger in die Daten 216, die sich auf spezifische interessierende Daten beziehen, im Wesentlichen gleich sein.

[0015] Da die Metadaten 214 und die Daten 216 zu einem einzelnen Journalblock 212 kombiniert werden können, können Journalschreibvorgänge in Segmenten oder Stapeln erfolgen und können die Daten und Metadaten zu einer einzelnen Schreiboperation kombiniert werden. Die Speicherung sowohl der Metadaten 214 als auch der Daten 216 in einem einzelnen Journalblock 212 kann daher eine ungefähr 50 %-ige Verringerung von separaten Schreiboperationen gegenüber dem Schreiben der Metadaten 214 und Daten 216 in unterschiedliche Stellen bieten.

[0016] Bei einigen Ausführungsformen kann das Journal 206 ein Rundjournal sein. Das heißt, dass der Cache 104 generell sequenziell in das Journal 206 schreiben kann und bei Erreichen des Endes des Journals 206 die nächste Schreiboperation ein Wrap-around zu einem Ausgangspunkt durchführen kann, um mit der nächsten Runde zu beginnen. Die Metadaten und Daten, die Schreib-Treffern an den gecachten Daten entsprechen, können in einen neuen Journalblock 212 in dem Journal 206 geschrieben werden. Durch das sequenzielle Schreiben kann ein Erfordernis vermieden werden, andernfalls in jedem Journalblock 212 gespeicherte Metadaten 214 lesen zu müssen. Eine Unterstützung für sequenzielle Schreibvorgänge kann jedoch auch bedeuten, dass das Journal 206 mehrere Journalblöcke 212 aufweist, die demselben Cache-Block entsprechen. Zum Beispiel kann ein erstes Schreiben an einer Speicheradresse 8 in einem Journalblock 1 nachverfolgt werden. Ein anschließendes Schreiben an derselben Speicheradresse, der Speicheradresse 8, kann in einem Journalblock 3 nachverfolgt werden (falls zum Beispiel der Cache 104 dazwischengeschaltete Cache-Block-Aktualisierungen verarbeitet hat, bei denen ein Journalblock 2 verwendet worden ist). Selbst wenn die Journalblöcke 1 und 2 auch Metadaten und Daten nachverfolgen, die der Speicheradresse 8 entsprechen, kann der Cache 104 Verarbeitungszeit für die bestehenden Journalblöcke einsparen. Stattdessen ermöglicht es die Auslegung des Journals 206 dem Cache 104, den Eintrag für den Journalblock 3 direkt in das Journal 206 zu schreiben, ohne dass weiter Metadaten gelesen werden müssen. Daher kann durch die sequenzielle Auslegung die Leistung verbessert werden.

[0017] Das Journal 206 kann ferner generell mehrere Speicherungsvorrichtungen unterstützen. Das heißt, dass das Journal 206 nicht zwischen Cache-Blöcken aus unterschiedlichen interessierenden gecachten Ziel-Speicherungsvorrichtungen unterscheidet. Diese Mehr-Laufwerk-Unterstützung kann generell zu einer besseren Raumausnutzung in dem Journal 206 führen, da durch die Mehr-Laufwerk-Unterstützung generell das Erfordernis der Vorreservierung von Raum für unterschiedliche Speicherungsvorrichtungen eliminiert wird. Andernfalls kann das Journal 206 ungenutzten Raum enthalten, der für eine Speicherungseinrichtung vorreserviert ist, die den Raum nicht benötigt, was zu einer ineffizienten Nutzung von Ressourcen führen kann.

[0018] Das Journal 206 ohne den Invalidationsdatenbereich 208 kann jedoch auch eine verringerte Leistung zeigen. Ein beispielhafter Verwendungsfall umfasst das Cachen von mehreren Speicherungsvorrichtungen und das Arbeiten in einem Zurückschreibmodus durch den Cache 104 (z. B. Aufschieben des Schreibens von aktualisierten Cache-Daten in die zugrundeliegende Speicherungseinrichtung). Falls eine Räumung aus dem Cache 104 in eine Speicherungseinrichtung fehlschlägt, kann das System keine neuen Daten cachen, nicht einmal neue Daten für andere Speicherungsvorrichtungen. Stattdessen kann das System die alten Daten bewahren, so dass sie in die Speicherungsvorrichtung zurückgeschrieben werden können. Bei der vorstehend beschriebenen protokollbasierten nichtflüchtigen Implementierung kann generell erwartet werden, dass die Räumung sequenziell erfolgt. Bei einigen Ausführungsformen kann der Cache 104 keine Daten wegen einer nicht zur Verfügung stehenden Speicherungsvorrichtung verwerfen, sofern der Benutzer nicht ausdrücklich einen anderen Befehl erteilt.

[0019] Selbst im Fall einer nicht zur Verfügung stehenden Speicherungsvorrichtung kann der Cache 104 jedoch fortfahren, I/O-Operationen zu pflegen, um einen transparenten Dienst für andere Speicherungsvorrichtungen, die immer noch zur Verfügung stehen, zu bieten. Dieses transparente Cachen kann wie folgt erreicht werden:

1) Bei einem Cache-Nichttreffer kann der Cache 104 die I/O-Operation durchleiten.

2) Bei einem Lese-Treffer kann der Cache 104 die angeforderte Leseoperation aus dem Cache 104 pflegen.

3) Bei einem Schreib-Treffer kann der Cache 104 die angeforderten Daten aus dem Cache entweder (a) aktualisieren oder (b) invalidieren.

[0020] Beide Operationen können zu einem Lese-Modifizier-Schreib-Zyklus für die Metadaten 214 und einer Schreiboperation für die Daten 216 führen (zum Beispiel im Fall einer Aktualisierungsanforderung). Somit kann jeder Schreib-Treffer 1 Lesen und 1 Schreiben (für eine Invalidierung) oder 2 Schreibvorgänge (für eine Aktualisierung) benötigen. Beide Szenarien können ein Gesamt-Leistungs-Penalty darstellen. Jede dieser Vorgehensweisen kann von der protokollbasierten Vorgehensweise wegführen, bei der ein Journal 206 ohne Invalidationsdatenbereich 208 zum Schreiben von Daten verwendet wird. Ferner können diese Szenarien das Risiko eines Datenverlustes beinhalten, da die Operationen nicht winzig klein sind und daraus Nutzen ziehen können, dass sie seriell durchgeführt werden.

[0021] **Fig.** 2B zeigt ein beispielhaftes Blockschaltbild eines Caches 104 gemäß einigen Ausführungsformen der vorliegenden Offenlegung. Der Cache 104 kann den Invalidationsdatenbereich 208 aufweisen. In dem Invalidationsdatenbereich 208 können generell Invalidationsaufzeichnungen 218 für Cache-Blöcke gespeichert werden, die aus dem Cache 104 gelöscht oder geräumt werden.

[0022] Der Invalidationsdatenbereich 208 kann eine separate Region des Caches 104 (z. B. getrennt von dem Journal 206) aufweisen. Das System kann unter Verwendung der Invalidationsaufzeichnungen 218 zugrundeliegende Journalblöcke in diese separate Region abbilden. Bei einigen Ausführungsformen kann der Cache die separate Region unter Verwendung eines zweckbestimmten vorbestimmten Namensraums implementieren.

[0023] Entsprechend kann der Invalidationsdatenbereich 208 folgende Vorteile bieten:

1) Er hält eine protokollbasierte Vorgehensweise zum Schreiben von Journaldaten bei. Das heißt, dass die Auslegung des Invalidationsdatenbereichs 208 Schreiboperationen, die andernfalls potenziell willkürliche Aktualisierungs- oder Invalidierungs-Schreibvorgänge sein können, in sequenzielle Schreibvorgänge an der Cache-Vorrichtung umwandeln kann.

2) Er bildet mehrere Journalblöcke in einen einzelnen Invalidationsaufzeichnungsblock ab (in **Fig.** 3A gezeigt). Zum Beispiel können bei einigen Ausführungsformen des Invalidationsdatenbereichs 208 drei Journalblöcke in eine Invalidationsaufzeichnung abgebildet werden. Folglich kann der Raum, der für den Invalidationsdatenbereich 208 verwendet wird, ungefähr 0,5 % einer Gesamtgröße des Caches 104 betragen.

3) Da die Größe des Invalidationsdatenbereichs 208 ein kleiner Bruchteil der Gesamtgröße des Caches 104 sein kann, kann der Invalidationsdatenbereich 208 generell insgesamt in einem RAM gespeichert werden. Ferner kann durch das generelle Speichern des Invalidationsdatenbereichs 208 im RAM kein Erfordernis bestehen, eine Leseoperation während des Invalidierens durchzuführen. Selbst wenn der Invalidationsdatenbereich nicht generell im RAM gespeichert ist, kann das System immer noch eine 66 %-ige Verringerung der Anzahl von erforderlichen Lesevorgängen aufweisen. Der Grund dafür ist, dass Aufzeichnungen für drei Journalblöcke in einen Invalidationsblock abgebildet werden können.

4) Durch das Packen von Invalidationsaufzeichnungsblock-Einträgen kann der Schreibaufwand so verringert werden, dass mehrere Einträge in einer Schreiboperation geschrieben werden. Ferner kann es eine 66 %g-ige Verringerung einer Anzahl von Schreibvorgängen geben.

[0024] Generell kann der Invalidationsdatenbereich 208 eine transparente Lösung für die Fehlerhandhabung und Aufrechterhaltung der Datenkonsistenz bieten. Ferner kann der Invalidationsdatenbereich 208 diese Vorteile bieten, ohne dass im Gegenzug ein großes Leistungs-Penalty eingetragen wird.

[0025] **Fig.** 3A zeigt ein beispielhaftes Abbilden zwischen dem Journal 206 und dem Invalidationsdatenbereich 208 gemäß einigen Ausführungsformen der vorliegenden Offenlegung. **Fig.** 3A zeigt das Journal 206 und den Invalidationsdatenbereich 208. Das Journal 206 weist Journalblöcke 1-3 auf. Der Invalidationsdatenbereich 208 weist eine Invalidationsaufzeichnung 1 auf. Die Invalidationsaufzeichnung 1 weist abgebildete Journalblöcke 1-3 auf.

[0026] Bei einigen Ausführungsformen können die Invalidationsaufzeichnungen generell in dem Cache 104 in dem separaten Invalidationsdatenbereich gespeichert sein. Die Invalidationsaufzeichnungen können generell abgebildete Journalblöcke verwenden, die einer Invalidationsaufzeichnung zugeordnet sind, um mehrere Journalblöcke darzustellen, die dem Journal 206 zugeordnet sind. Zum Beispiel kann der Journalblock 1 dem abgebildeten Journalblock 1 entsprechen, kann der Journalblock 2 dem abgebildeten Journalblock 2 entsprechen und kann der Journalblock 3 dem abgebildeten Journalblock 3 entsprechen. Ferner können die abgebildeten Journalblöcke 1-3 weniger zu speichernde Metadaten benötigen als die entsprechenden

zugrundeliegenden Journalblöcke 1-3. Entsprechend kann bei einigen Ausführungsformen das System einen Teilsatz der Metadaten der zugrundeliegenden Journalblöcke 1-3 auswählen, so dass alle drei Journalblöcke in der Invalidationsaufzeichnung 1 gespeichert werden können.

[0027] **Fig.** 3B zeigt ein weiteres beispielhaftes Abbilden zwischen dem Journal 206 und dem Invalidationsdatenbereich 208 gemäß einigen Ausführungsformen der vorliegenden Offenlegung. **Fig.** 3B umfasst das Journal 206 und den Invalidationsdatenbereich 208. Das Journal 206 weist den Journalblock 1 mit den Metadaten 214 und den Daten 216 auf. Der Invalidationsdatenbereich 208 weist die Invalidationsaufzeichnung 1 auf. Die Invalidationsaufzeichnung 1 weist den abgebildeten Journalblock 1 auf. Der abgebildete Journalblock 1 weist Metadaten 302 auf.

[0028] Die Invalidationsaufzeichnung 1 kann sowohl eine in einem Cache gespeicherte Version und eine relativ schnellere Version, die in einen Schreib/Lesespeicher (random access memory - RAM) geladen ist, aufweisen. Die in-RAM-Datenstruktur kann generell die Leistung verbessern und das Erfordernis verringern, Daten aus dem relativ langsameren Journal oder aus dem Cache 104 zu lesen. Bei einigen Ausführungsformen kann eine beispielhafte Definition der Invalidationsaufzeichnung 1 das Folgende umfassen:

```
/* As stored on cache 104 */

struct invalidation_record_block {

        unsigned char checksum[16]; // checksum of the entire block with salt

        struct mapped_journal_block journal_block[3];

}
```

[0029] Eine beispielhafte Invalidationsaufzeichnung kann mehrere abgebildete Journalblöcke („journal_block“) und einen Fehlerkorrekturcode („checksum“) aufweisen.

[0030] Bei einigen Ausführungsformen der Invalidationsaufzeichnungs-Datenstruktur kann eine beispielhafte Definition des abgebildeten Journalblocks, auf den in der Invalidationsaufzeichnungs-Datenstruktur Bezug genommen wird, umfassen:

```
struct mapped_journal_block {

        unsigned long long epoch; // journal epoch during the time of writing.

        unsigned int target_lba[MAX_JOURNAL_ENTRY]; // store offsets (4k
aligned LBA's)

}
```

[0031] Der abgebildete Journalblock kann eine Ansammlung (z.B. ein Array) von Speicheradressen und Offsets („target_lba“) aufweisen. Die Speicheradressen können einen interessierenden Speicherblock identifizieren, und der Offset kann spezifische interessierende Speicherblöcke oder spezifische interessierende Speicherstellen innerhalb der Speicherblöcke identifizieren. Die Ansammlung von Speicheradressen und Offsets in dem abgebildeten Journalblock kann auf eine entsprechende Ansammlung von Speicheradressen und Offsets, die in den zugrundeliegenden Journalblöcken gespeichert sind, abgebildet werden. Der abgebildete Journalblock kann ferner einen Zeitstempel („epoch“) aufweisen, der mit einem entsprechenden Zeitstempel übereinstimmen kann, welcher in dem zugrundeliegenden Journalblock gespeichert ist.

[0032] Bei einigen Ausführungsformen kann die in-RAM-Datenstruktur, die eine Invalidationsaufzeichnung darstellt, das Folgende umfassen.

```
/* In RAM representation of invalidation record block */

struct in_ram_invalidation_record_block {

        unsigned char valid:1; // whether this block has been written to before

        unsigned char outstanding:1; // Max value can only be 1 at any time.

        void *waiting_creqs; // Waiting requests if there's already an outstanding

write

        void *irb_cache_device; // buffer containing the IRB on the cache device

}
```

[0033] Die in-RAM-Datenstruktur kann generell die Leistung verbessern und ein Erfordernis zum Lesen von Daten aus dem relativ langsameren Journal oder aus dem Cache 104 verringern.

[0034] Der abgebildete Journalblock kann einen Journalblock darstellen. Bei einigen Ausführungsformen kann eine Invalidationsaufzeichnung mehrere abgebildete Journalblöcke aufweisen. Zum Beispiel zeigt **Fig.** 3A eine Invalidationsaufzeichnung mit einer Kapazität von drei abgebildeten Journalblöcken (so dass es eine 3-zu-1-Abbildung aus Journalblöcken in eine Invalidationsaufzeichnung geben kann). Der abgebildete Journalblock kann Einträge von Speicheradressen, die invalidiert worden sind, in dem Cache 104 speichern. Bei einigen Ausführungsformen können die Speicheradressen Logikblockadressen (LBAs) sein. Obwohl die vorliegende Offenlegung das Nachverfolgen von drei Journalblöcken unter Verwendung einer einzelnen Invalidationsaufzeichnung beschreibt, kann die Invalidationsaufzeichnung jede Anzahl von abgebildeten Journalblöcken aufweisen, zum Beispiel ermittelt auf der Basis des Teilsatzes von Metadaten, die zum Speichern in dem abgebildeten Journalblock gewählt worden sind. Ein beispielhafter Journalblock kann ungefähr 256 kB groß sein, und eine beispielhafte Invalidationsaufzeichnung kann ungefähr 4 kB groß sein. Da es eine 3-zu-1-Abbildung zwischen den Journalblöcken und den Invalidationsaufzeichnungen geben kann, kann der Invalidationsdatenbereich 208 raumeffizient sein. Zum Beispiel kann der Invalidationsdatenbereich 208 nur ungefähr 4 kB nutzen, um 768 kB (3 Journalblöcke x 256 kB pro Journalblock) Daten in dem Journal 206 zu ergeben. Entsprechend kann der Raumbedarf für den Invalidationsdatenbereich 208 ungefähr 0,52 % (4 kB/768 kB) betragen. Ferner kann durch die 3-zu-1-Abbildung zwischen den abgebildeten Journalblöcken und den Invalidationsaufzeichnungen die Anzahl von Lese- und Schreiboperationen, die während des Invalidierungsprozesses durchgeführt werden, um ungefähr 66 % verringert werden. Bei einigen Ausführungsformen kann aufgrund der kleinen Größe des Invalidationsdatenbereichs 208 und der effizienten Raumzuteilung der gesamte Invalidationsdatenbereich 208 in dem Schreib-/Lesespeicher (RAM) gespeichert werden, um die Anzahl von Leseoperationen in dem Cache 104 zu verringern oder sogar vollständig zu eliminieren.

[0035] Bei einigen Ausführungsformen kann der Invalidationsdatenbereich 208 einen Teilsatz der Metadaten 214, die in dem Journal 206 nachverfolgt werden, speichern. Diese Effizienz kann ebenfalls zu der kleinen Größe des Invalidationsdatenbereichs 208 beitragen. Zum Beispiel können die in dem Journal 206 nachverfolgten Metadaten 214 eine Speicheradresse (z. B. eine Logikblockadresse (LBA)), einen Cache-Block-Typ, einen Offset und einen Hash-Wert für eine Fehlerkorrektur aufweisen. Im Gegensatz dazu können bei einigen Ausführungsformen die Metadaten 302, die in dem Invalidationsdatenbereich 208 nachverfolgt werden, einen Teilsatz der Metadaten 214 aufweisen, die in dem Journal 206 nachverfolgt werden. Zum Beispiel kann das System wählen, nur eine entsprechende Speicheradresse in den Metadaten 302 nachzuverfolgen. Durch das Nachverfolgen nur eines Teilsatzes der Metadaten kann die Raumeffizienz oder Kapazität des Invalidationsdatenbereichs 208 verbessert werden.

[0036] Weitere Modifikationen, die abhängig sind von der Verwendung für das/den und von in dem Journal 206 und dem Invalidationsdatenbereich 208 gespeicherten Metadaten können ferner diese Größe beeinflussen. Beispiele für Modifikationen können das Erhöhen der Größe der Journalblöcke, Verringern der Größe der Speicheradressen, die in einem abgebildeten Journalblock gespeichert sind, etc. umfassen. Bei einigen Ausführungsformen des Systems kann die Größe der Speicheradressen, die in einem abgebildeten Journalblock gespeichert sind, verkürzt sein. Bei einer Implementierung kann die Verkürzung auf einer Speicher-

größe der zugrundeliegenden Speicherungsvorrichtung basieren. Zum Beispiel kann, falls die Speicherungsvorrichtung ausreichend klein ist, das System die Speicheradressen von ungefähr vier Bytes in dem abgebildeten Journalblock speichern im Vergleich zu einer vollständigen Speicheradresse von ungefähr acht Bytes, die in einem entsprechenden Journalblock gespeichert ist.

[0037] Bei einer weiteren Implementierung kann die Verkürzung das Ermitteln eines Offsets auf der Basis einer Speicheradresse, die in der zugrundeliegenden Speicherungseinrichtung gespeichert ist, und das Speichern des Offsets anstelle der Speicheradresse umfassen. Bei Ausführungsformen des Caches können Datenblöcke mit Größen von ungefähr 4 kB gespeichert werden. (Falls eine I/O-Anforderung eine kleinere Größe betrifft, kann der Cache die verbleibenden Daten, die dem Datenblock aus der Speicherungseinrichtung zugeordnet sind, aufrufen und den gesamten Inhalt des 4-kB-Datenblocks cachen.) Folglich kann bei einigen Ausführungsformen die Verkürzung das Umwandeln einer Speicheradresse (wie z. B. einer LBA) der zugrundeliegenden Speicherungseinrichtung in Offsets umfassen. Bei einigen Ausführungsformen können die Offsets ungefähr 4 kB umfassen. Zum Beispiel kann ein Offset 0 die ersten 4 kB an der Speicherungsvorrichtung darstellen, kann ein Offset 1 die nächsten 4 kB an der Speicherungseinrichtung darstellen etc. Entsprechend kann der Cache eine Speicheradresse von zum Beispiel einer 512-Byte-LBA in eine als nächstes verfügbare ausgerichtete 4-kB-LBA umwandeln. Statt eine vollständige LBA zu speichern, kann das System eine vollständige LBA in einen Offset umwandeln, der eine kleinere Anzahl von Bytes verwendet, und den Offset in der Invalidationsaufzeichnung und dem abgebildeten Journalblock speichern. Zum Beispiel kann eine LBA 0-7 in der zugrundeliegenden Speicherungseinrichtung einer LBA 0 in dem Cache zusammen mit einem optionalen Offset entsprechen. Bei einigen Ausführungsformen des Invalidationsdatenbereichs kann ein Offset-Feld mit einer Größe von 4 B dadurch bis zu 16 Terabytes der zugrundeliegenden Speicherungseinrichtung ($2^{32} \times 4.096$) adressieren. Bei größeren Speicherungsvorrichtungen kann bei einigen Ausführungsformen des Systems die Cache-Block-Größe auf ungefähr 8 kB oder mehr erhöht werden, die Offset-Größe auf ungefähr 5 Bytes erhöht werden etc.

[0038] Bei einigen Ausführungsformen des Caches 104 können Cache-Blöcke durch Ermitteln einer Abbildung zwischen dem Journal 206 und dem Invalidationsdatenbereich 208 invalidiert werden. Das heißt, dass der Cache 104 eine geeignete Invalidationsaufzeichnung, einen abgebildeten Journalblock und einen entsprechenden Index in dem Invalidationsdatenbereich 208 für einen Datenblock auf der Basis des Journalblocks und des Indexes in dem Journal 206 ermitteln kann.

[0039] Zum Beispiel sei angenommen, dass der Cache 104 eine Invalidierung eines Datenblocks, der sich in dem Journalblock 1 bei Index 3 (304a) befindet, durchführt. Auf der Basis des Journalblocks und des Indexes in dem Journal 206 kann der Cache 104 generell die entsprechende Invalidationsaufzeichnung, den abgebildeten Journalblock und den Index in dem abgebildeten Journalblock ermitteln. Zuerst kann der Cache 104 eine Invalidationsaufzeichnung auf der Basis des entsprechenden Journalblocks ermitteln. Da die Journalblöcke eine Abbildung von 3-zu-1 auf die Kapazität der Invalidationsaufzeichnungen durchführen können, können bei einigen Ausführungsformen des Caches 104 eine Divisionsoperation und eine Rundungsoperation (z. B. Aufrundung) durchgeführt werden, um die entsprechende Invalidationsaufzeichnung zu ermitteln. Zum Beispiel kann für den Journalblock 1 das System 1 / 3 = 0,33... und $\lceil 0{,}33... \rceil = 1$ berechnen, wodurch der Journalblock 1 auf die Invalidationsaufzeichnung 1 abgebildet wird. Bei einem weiteren Beispiel ist, falls das System den Journalblock 5 auf die Invalidationsaufzeichnung abbildet, 5 / 3 = 1,66... und $\lceil 1{,}66... \rceil = 2$, wodurch der Journalblock 5 auf die Invalidationsaufzeichnung 2 abgebildet wird.

[0040] Als Nächstes kann der Cache 104 einen abgebildeten Journalblock auf der Basis eines Journalblocks ermitteln. Bei einigen Ausführungsformen kann der Cache 104 eine Modulo-Operation anwenden, um einen abgebildeten Journalblock auf der Basis der Journalblocknummer zu ermitteln. Zum Beispiel kann für den Journalblock 1 das System 1 mod 3 = 1 berechnen, wodurch der Journalblock 1 auf den abgebildeten Journalblock 1 abgebildet wird. Auf im Wesentlichen gleiche Weise ist, falls das System den Journalblock 5 auf einen abgebildeten Journalblock 5 abbildet, 5 mod 3 = 2, wodurch der Journalblock 5 auf einen abgebildeten Journalblock 2 innerhalb der Invalidationsaufzeichnung 2 (die zuvor ermittelt worden ist) abgebildet wird.

[0041] Zuletzt kann der Cache 104 einen Index in dem abgebildeten Journalblock ermitteln, der einem Index in dem zugrundeliegenden Journalblock entspricht. Bei einigen Ausführungsformen des Caches 104 kann in dem abgebildeten Journalblock der gleiche Index verwendet werden wie der Index, der in dem zugrundeliegenden Journalblock verwendet wird. Das heißt, dass beim Schreiben der entsprechenden Journalblockeinträge in den Invalidationsdatenbereich 208 die Journalblockeinträge an dem gleichen Index in dem target_lba-Array des abgebildeten Journalblocks hinzugefügt werden können wie bei einem entsprechenden

target_lba-Array des Journalblocks. Entsprechend kann der Index in dem abgebildeten Journalblock leicht und schnell auf der Basis des Indexes in dem zugrundeliegenden Journalblock ermittelt werden.

[0042] Um ein umgekehrtes Abbilden (d. h. Ermitteln eines Journalblocks und eines entsprechenden Indexes auf der Basis einer Invalidationsaufzeichnung, eines abgebildeten Journalblocks und Indexes) durchzuführen, kann der Cache 104 umgekehrte Operationen zu den oben beschriebenen durchführen. Zum Beispiel kann der Cache 104 Informationen über eine Speicheradresse für einen invalidierten Cache-Block auf der Basis der Metadaten 302, die in dem abgebildeten Journalblock gespeichert sind, identifizieren. Der Cache 104 kann die Journalblocknummer auf der Basis der Invalidationsaufzeichnungsnummer ermitteln, und der Index für den Journalblock kann aus dem für den abgebildeten Journalblock verwendeten Index konkludiert werden.

[0043] **Fig.** 4 zeigt ein beispielhaftes Verfahren 400 zum Invalidieren unter Verwendung des Invalidationsdatenbereichs gemäß einigen Ausführungsformen der vorliegenden Offenlegung. Bei einigen Ausführungsformen kann das Verfahren 400 umfassen: Ermitteln eines Journalblocks für eine Speicheradresse in einer empfangenen Schreiboperation (Schritt 402); Ermitteln eines abgebildeten Journalblocks und eines Offsets auf der Basis des ermittelten Journalblocks und einer entsprechenden Invalidationsaufzeichnung aus dem Invalidationsdatenbereich (Schritt 404); Ermitteln, ob es ausstehende Schreiboperationen gibt (Schritt 406); falls ja, Vereinigen der Schreiboperationen und Durchführen der Schreiboperationen als einzelnes Schreiben in den Cache (Schritt 408); falls nein, Durchführen der empfangenen Schreiboperation (Schritt 410).

[0044] Zuerst wird bei dem Verfahren 400 ein Journalblock für eine Speicheradresse in einer empfangenen Schreiboperation ermittelt (Schritt 402). Bei einigen Ausführungsformen des Verfahrens 400 kann der Journalblock auf der Basis der Logikblockadresse (LBA) in der empfangenen Schreiboperation identifiziert werden. Oder bei einem Schreib-Treffer (was bedeutet, dass die LBA zuvor gecacht worden ist) kann bei dem Verfahren 400 der Journalblock auf der Basis der LBA identifiziert werden, an der der Cache-Block in dem Cache gespeichert ist.

[0045] Dann geht das Verfahren 400 zum Ermitteln eines abgebildeten Journalblocks auf der Basis des ermittelten Journalblocks und einer entsprechenden Invalidationsaufzeichnung aus dem Invalidationsdatenbereich weiter (Schritt 404). Bei einigen Ausführungsformen kann die Invalidationsaufzeichnung durch Durchführen von Divisionsoperationen und Rundungsoperationen an der Journalblocknummer ermittelt werden. Bei einigen Ausführungsformen kann ferner der Index für den abgebildeten Journalblock unter Verwendung des in dem zugrundeliegenden Journalblock verwendeten Indexes ermittelt werden. Zum Beispiel kann dann, wenn das System die gleichen Indizes für den abgebildeten Journalblock und den zugrundeliegenden Journalblock verwendet, der Index leicht und schnell ermittelt werden.

[0046] Dann kann bei dem Verfahren 400 ermittelt werden, ob es ausstehende Schreiboperationen gibt (Schritt 406). Bei einigen Ausführungsformen kann dieses Ermitteln unter Verwendung einer in-RAM-Datenstruktur erfolgen, die der Invalidationsaufzeichnung entspricht. Zum Beispiel kann die in-RAM-Datenstruktur ein Feld („ausstehend“) enthalten, das identifiziert, ob Schreiboperationen ausstehen. Ein Vorteil der Verwendung der in-RAM-Datenstruktur besteht in dem Vermeiden einer relativ langsameren Leseoperation in dem zugrundeliegenden Cache zum Aufrufen der gespeicherten Invalidationsaufzeichnung.

[0047] Falls bei dem Verfahren 400 ermittelt wird, dass Schreiboperationen ausstehen (Schritt 406: Ja), können die Schreiboperationen und das Durchführen der Schreiboperationen zu einem einzelnen Schreiben in den Cache vereinigt werden (Schritt 408). Bei einigen Ausführungsformen des Verfahrens 400 werden anschließende Schreibvorgänge bei einer Ermittlung, dass Schreiboperationen ausstehen, in eine Warteschlange eingereiht. Wenn die vorhergehende Schreiboperation abgeschlossen ist, werden bei dem Verfahren 400 die in der Warteschlange befindlichen Schreibvorgänge als einzelnes Schreiben, das die vereinigten Informationen sämtlicher Aktualisierungen enthält, in den Cache geschrieben. Bei einigen Ausführungsformen kann das Vereinigen das Identifizieren von Schreiboperationen, die an demselben Datenblock erfolgen, das Ordnen der Schreiboperationen auf der Basis eines Zeitstempels und das Ermitteln des Endergebnisses der geordneten Schreiboperationen umfassen. Auf diese Weise ermöglicht dieses Stapeln oder Vereinigen von Schreiboperationen, dass bei dem Verfahren 400 die Anzahl von Lese- und Schreiboperationen, die für die Invalidierung verwendet werden, weiter verringert werden kann. Falls ermittelt wird, dass keine Schreiboperationen ausstehen (Schritt 406: Nein), kann das Verfahren 400 zum Durchführen der empfangenen Schreiboperation weitergehen (Schritt 410).

[0048] Fig. 5 zeigt ein beispielhaftes Verfahren 500 für eine Cache-Wiederherstellung gemäß einigen Ausführungsformen der vorliegenden Offenlegung. Die Cache-Wiederherstellung bezieht sich auf eine Situation, in der dem Cache die Rekonstruktion auf der Basis des Journals und des Invalidationsdatenbereichs zugutekommen kann, zum Beispiel nach einem Energieausfall, einer fehlerhaften Systemabschaltung oder einem anderen unerwarteten Ereignis. Das Verfahren 500 kann das Rekonstruieren des Caches auf der Basis des Journals (Schritt 502) umfassen; dann für jeden abgebildeten Journalblock in jeder Invalidationsaufzeichnung (Schritt 504): Ermitteln auf der Basis des Abgebildet-Journal-Eintrags, ob ein entsprechender Cache-Block valid ist (Schritt 506); falls ja, Rückkehr zu Schritt 504, falls nein, Räumen des veralteten Blocks aus dem Cache (Schritt 508).

[0049] Zuerst wird bei dem Verfahren 500 der Cache auf der Basis des Journals rekonstruiert (Schritt 502). Bei einigen Ausführungsformen des Systems kann angenommen werden, dass der Inhalt des Journals generell valide Daten darstellt, die in dem Cache rekonstruiert werden sollen. Bei einigen Ausführungsformen kann das System den Cache durch Aufrufen jedes Journalblocks aus dem Journal und iteratives Verarbeiten der Metadaten in jedem Journalblock zum Rekonstruieren jedes Cache-Blocks rekonstruieren. Die Journalblock-Metadaten können jedoch Cache-Blöcke enthalten, die invalidiert worden sein können. Das System kann später diese anfängliche Annahme von generell validen Daten auf der Basis des Invalidationsdatenbereichs korrigieren. Zum Beispiel kann das System nichtvalide Cache-Blöcke auf der Basis des Invalidationsdatenbereichs identifizieren und diese veralteten Cache-Blöcke aus dem Cache räumen.

[0050] Als Nächstes iteriert das Verfahren 500 durch jeden abgebildeten Journalblock in jeder Invalidationsaufzeichnung (Schritt 504). Für jeden abgebildeten Journalblock werden die Metadaten in dem abgebildeten Journalblock verarbeitet, um zu ermitteln, ob die entsprechenden Cache-Blöcke valid oder nichtvalid sind (Schritt 506). Bei einigen Ausführungsformen kann die Ermittlung, ob ein Cache-Block valid ist, durch Ermitteln erfolgen, ob die Metadaten in dem abgebildeten Journalblock mit den zugrundeliegenden Metadaten in dem zugrundeliegenden Journalblock konsistent sind. Zum Beispiel kann die Konsistenz der zugrundliegenden Metadaten in dem zugrundeliegenden Journalblock durch Lokalisieren der entsprechenden Journalblocknummer und des Indexes auf der Basis von Divisionsoperationen, Rundungsoperationen und Modulo-Operationen ermittelt werden. Dann können die Metadaten, die an der ermittelten Journalblocknummer und dem ermittelten Index gespeichert sind, mit den Metadaten aufbereitet werden, die in dem abgebildeten Journalblock gespeichert sind. Zum Beispiel sei angenommen, dass bei dem Verfahren 500 auf der Basis des abgebildeten Journalblocks identifiziert wird, dass erwartet wird, dass die Speicheradresse 8 an dem zugrundeliegenden Journalblock 1, Index 3 zu finden ist. Bei dem Verfahren 500 kann dann der entsprechende Inhalt des Journalblocks 1 bei Index 3 der Metadaten aufgerufen werden. Falls dieser Journalblock einen Cache-Block, der der Speicheradresse 8 entspricht, nachverfolgt, kann ermittelt werden, dass der Cache-Block, der der Speicheradresse 8 entspricht, nichtvalid ist, da der erwartete Cache-Block, der auf dem Invalidationsdatenbereich und dem abgebildeten Journalblock basiert, mit dem eigentlichen Cache-Block übereinstimmt, der in dem entsprechenden zugrundeliegenden Journalblock nachverfolgt wird. Andererseits sei angenommen, dass bei dem Verfahren 500 auf der Basis des abgebildeten Journalblocks identifiziert wird, dass erwartet wird, dass ein Cache-Block, der einer Speicheradresse 16 entspricht, an dem zugrundeliegenden Journalblock 1, Index 4 zu finden ist. Falls der eigentliche Cache-Block, der an dem zugrundeliegenden Journalblock 1, Index 4 gespeichert ist, nicht mit der Speicheradresse 16 übereinstimmt, kann das Verfahren 500 zum Verarbeiten der nächsten Metadaten weitergehen, da der Cache-Block in dem Cache verbleiben kann, wenn der erwartete Cache-Block, der auf dem Invalidationsdatenbereich und dem abgebildeten Journalblock basiert, nicht mit dem eigentlichen Cacheblock übereinstimmt, der in dem entsprechenden zugrundeliegenden Journalblock nachverfolgt wird.

[0051] Falls die Metadaten übereinstimmen, kann bei dem Verfahren 500 ermittelt werden, dass der Cache-Block nichtvalid ist (Schritt 506: Nein). Entsprechend kann bei dem Verfahren 500 der nichtvalide (d. h. veraltete) Block aus dem Cache geräumt oder verworfen werden (Schritt 508). Falls die Metadaten nicht übereinstimmen, kann das Verfahren 500 zum Verarbeiten der nächsten Megadaten, die dem abgebildeten Journalblock entsprechen, weitergehen oder zum Verarbeiten des nächsten abgebildeten Journalblocks weitergehen, falls bei dem Verfahren 500 sämtliche Metadaten in dem abgebildeten Journalblock verarbeitet worden sind (Schritt 506: Ja).

[0052] Der Invalidationsdatenbereich kann ferner einige weitere Vorteile bieten bezüglich (1) transparenten Cachens und (2) Dynamisch-Cache-Modus-Umschaltens zwischen einem Zurückschreib- und einem Durchgängigschreib-Modus. Das transparente Cachen bezieht sich auf eine Fähigkeit eines Administrators oder Nutzers, den Cache nach Belieben aus dem System zu entfernen. Das Dynamik-Cache-Modus-Umschalten bezieht sich auf eine Fähigkeit eines Administrators oder Nutzers, den Cache-Modus zwischen einem

Zurückschreib- und einem Durchgängigschreib-Modus umzuschalten, ohne dass das System abgeschaltet werden muss. Der Invalidationsdatenbereich kann ein transparentes Cachen und Dynamik-Cache-Modus-Umschalten ermöglichen, ohne dass eine signifikante Latenz in laufende I/O-Operationen eingetragen wird. Bei einigen Ausführungsformen kann bei dem Cache eine Latenz durch Verwerfen sämtlicher Daten vermieden werden. Falls sich der Cache im Zurückschreib-Modus befindet, entleert der Cache generell seine veränderten Daten in die zugrundeliegende Speicherungsvorrichtung (d. h. „schreibt“ die Daten „zurück“), bevor der Cache die Daten verwerfen oder räumen kann. Zuvor hat der Cache seine Daten durch Pausierenlassen sämtlicher ausstehender I/O-Operationen vor dem Entleeren entfernt. Durch das Anhalten oder Pausierenlassen sämtlicher ausstehender I/O-Operationen kann jedoch eine unerwünschte Latenz eingetragen werden, da es keine Obergrenze für den Zeitraum gibt, in dem ein Entleeren stattfindet. Beispielhafte Faktoren, die die Entleerungszeit beeinflussen können, können die Menge an veränderten Daten, Zufälligkeit, Plattengeschwindigkeit etc. umfassen. Bei einigen Ausführungsformen verbessert der Invalidationsdatenbereich laufende I/O-Operationen durch Setzen des Caches in den Pausiermodus und Pflegen von I/O-Operationen wie folgt:

1) Der Cache leitet Cache-Nichttreffer durch

2) Der Cache pflegt Lese-Treffer

3) Der Cache verwendet den Invalidationsdatenbereich zum Invalidieren von Schreib-Treffern und leitet die Schreibvorgänge zu der zugrundeliegenden Speicherungseinrichtung durch.

[0053] Wenn die Entleerung des Caches beendet ist, kann der Cache sämtliche Daten sicher verwerfen.

[0054] Fachleute auf dem Sachgebiet erkennen, dass verschiedene Darstellungen, die hier beschrieben worden sind, als elektronische Hardware, Computersoftware oder Kombinationen aus beiden implementiert werden können. Zur Veranschaulichung dieser Austauschbarkeit von Hardware und Software sind oben verschiedene veranschaulichende Blöcke, Module, Elemente, Komponenten, Verfahren und Algorithmen generell hinsichtlich ihrer Funktionalität beschrieben worden. Ob eine solche Funktionalität als Hardware, Software oder eine Kombination implementiert wird, hängt von den besonderen Anwendungs- und Auslegungseinschränkungen ab, die dem Gesamtsystem auferlegt sind. Fachleute können die beschriebene Funktionalität auf verschiedene Arten für jede besondere Anwendung implementieren. Verschiedene Komponenten und Blöcke können unterschiedlich angeordnet (zum Beispiel in einer anderen Reihenfolge angeordnet oder auf eine andere Art unterteilt) sein, ohne dass dadurch vom Schutzbereich der vorliegenden Technologie abgewichen wird.

[0055] Ferner kann eine Implementierung des Invalidationsdatenbereichs zentralisiert in einem Computersystem oder verteilt realisiert werden, wobei verschiedene Elemente über mehrere miteinander verbundene Computersysteme verteilt sind. Jede Art von Computersystem oder anderer Einrichtung, die zum Ausführen der hier beschriebenen Verfahren vorgesehen ist, ist zum Durchführen der hier beschriebenen Funktionen geeignet.

[0056] Eine typische Kombination aus Hardware und Software kann ein Universal-Computersystem mit einem Computerprogramm sein, das bei Ladung und Ausführung das Computersystem so steuert, dass dieses die hier beschriebenen Verfahren durchführt. Die Verfahren können ferner in ein Computerprogrammprodukt eingebettet sein, das sämtliche der Merkmale aufweist, die das Implementieren der hier beschriebenen Verfahren ermöglicht, und das bei Laden in ein Computersystem in der Lage ist, diese Verfahren durchzuführen.

[0057] Computerprogramm oder Anwendung im vorliegenden Kontext bedeutet jeden Ausdruck in jeder Sprache, Code oder Schreibweise eines Satzes von Befehlen zum Bewirken, dass ein System mit einer Informationsverarbeitungsfähigkeit eine besondere Funktion entweder direkt oder nach einem oder beidem des Folgenden durchführt: a) Umwandeln in eine andere Sprache, Code oder Schreibweise; b) Reproduzieren in einer anderen materiellen Form. Bezeichnenderweise können die hier beschriebenen Systeme und Verfahren ferner in anderen spezifischen Formen ausgeführt sein, ohne dass dadurch vom Wesen oder von wesentlichen Attributen derselben abgewichen wird, und entsprechend sollte hinsichtlich des Anzeigens des Schutzbereiches der Systeme und Verfahren auf die nachfolgenden Patentansprüche statt auf die vorstehende Beschreibung Bezug genommen werden.

Patentansprüche

1. Cache (104), der umfasst:
ein Journal (206), das zum Nachverfolgen von in dem Cache (104) gespeicherten Datenblöcken (216) ausgelegt ist; und
einen Invalidationsdatenbereich (208), der zum Nachverfolgen von invalidierten Datenblöcken ausgelegt ist, die den Datenblöcken (216) zugeordnet sind, welche in dem Journal (206) nachverfolgt werden, wobei sich der Invalidationsdatenbereich (208) in einer von dem Journal (206) getrennten Region des Caches (104) befindet.

2. Cache (104) nach Anspruch 1,
wobei das Journal (206) zum Nachverfolgen von Metadaten (214) für die Datenblöcke (216) ausgelegt ist und wobei die Metadaten (214) eine Speicheradresse umfassen, die dem Datenblock (216) entspricht, und
wobei der Invalidationsdatenbereich (208) zum Nachverfolgen von Metadaten (302) ausgelegt ist, die den invalidierten Datenblöcken zugeordnet sind, und wobei die zugeordneten Metadaten (302) eine Speicheradresse umfassen, die dem invalidierten Datenblock entspricht.

3. Cache (104) nach Anspruch 2,
wobei das Journal (206) zum Nachverfolgen der Datenblöcke (216) unter Verwendung von Journalblöcken (212) ausgelegt ist, wobei die Journalblöcke (212) zum Speichern der Metadaten (214) für die Datenblöcke (216) ausgelegt sind; und
wobei der Invalidationsdatenbereich (208) zum Nachverfolgen der Metadaten (302), die den invalidierten Datenblöcken zugeordnet sind, unter Verwendung von Invalidationsaufzeichnungen und abgebildeten Journalblöcken ausgelegt ist, wobei die abgebildeten Journalblöcke zum Speichern der zugeordneten Metadaten (302) für die invalidierten Datenblöcke ausgelegt sind und wobei die Invalidationsaufzeichnungen zum Speichern der abgebildeten Journalblöcke ausgelegt sind.

4. Cache (104) nach Anspruch 3,
wobei die Metadaten (214), die in dem Journal (206) nachverfolgt werden, ferner einen Index (304a) in eine Ansammlung von Metadaten (214) umfassen, die in jedem Journalblock (212) gespeichert sind, und
wobei die Metadaten (302), die in dem Invalidationsdatenbereich (208) nachverfolgt werden, ferner einen Index (304b) in eine Ansammlung von Metadaten (302) umfassen, die in jedem abgebildeten Journalblock gespeichert sind.

5. Cache (104) nach Anspruch 3, wobei der Cache (104) so ausgelegt ist, dass er eine Invalidationsaufzeichnungsnummer, die einer Invalidationsaufzeichnung in dem Invalidationsdatenbereich (208) zugeordnet ist, auf der Basis einer entsprechenden einem Journalblock zugeordneten Journalblocknummer ermittelt.

6. Cache (104) nach Anspruch 3, wobei der Cache (104) so ausgelegt ist, dass er eine Abgebildet-Journalblock-Nummer, die einem abgebildeten Journalblock in dem Invalidationsbereich (208) zugeordnet ist, auf der Basis einer entsprechenden einem Journalblock zugeordneten Journalblocknummer ermittelt.

7. Cache (104) nach Anspruch 4, wobei der Index (304a), der in dem Journal (206) nachverfolgt wird, so ausgewählt ist, dass er den gleichen Wert aufweist wie der Index (304b), der in dem Invalidationsdatenbereich (208) nachverfolgt wird.

8. Cache (104) nach Anspruch 1,
wobei die Speicheradresse, die in dem Invalidationsdatenbereich (208) nachverfolgt wird, verkürzt ist im Vergleich zu der Speicheradresse, die in dem Journal (206) nachverfolgt wird, und
wobei die Verkürzung auf der Basis von mindestens einem einer Speicherungsgröße einer zugrundeliegenden gecachten Speicherungsvorrichtung und eines Offsets ermittelt wird, der auf der Basis einer Speicheradresse eines Blocks in der zugrundeliegenden Speicherungsvorrichtung ermittelt wird.

Es folgen 7 Seiten Zeichnungen

Anhängende Zeichnungen

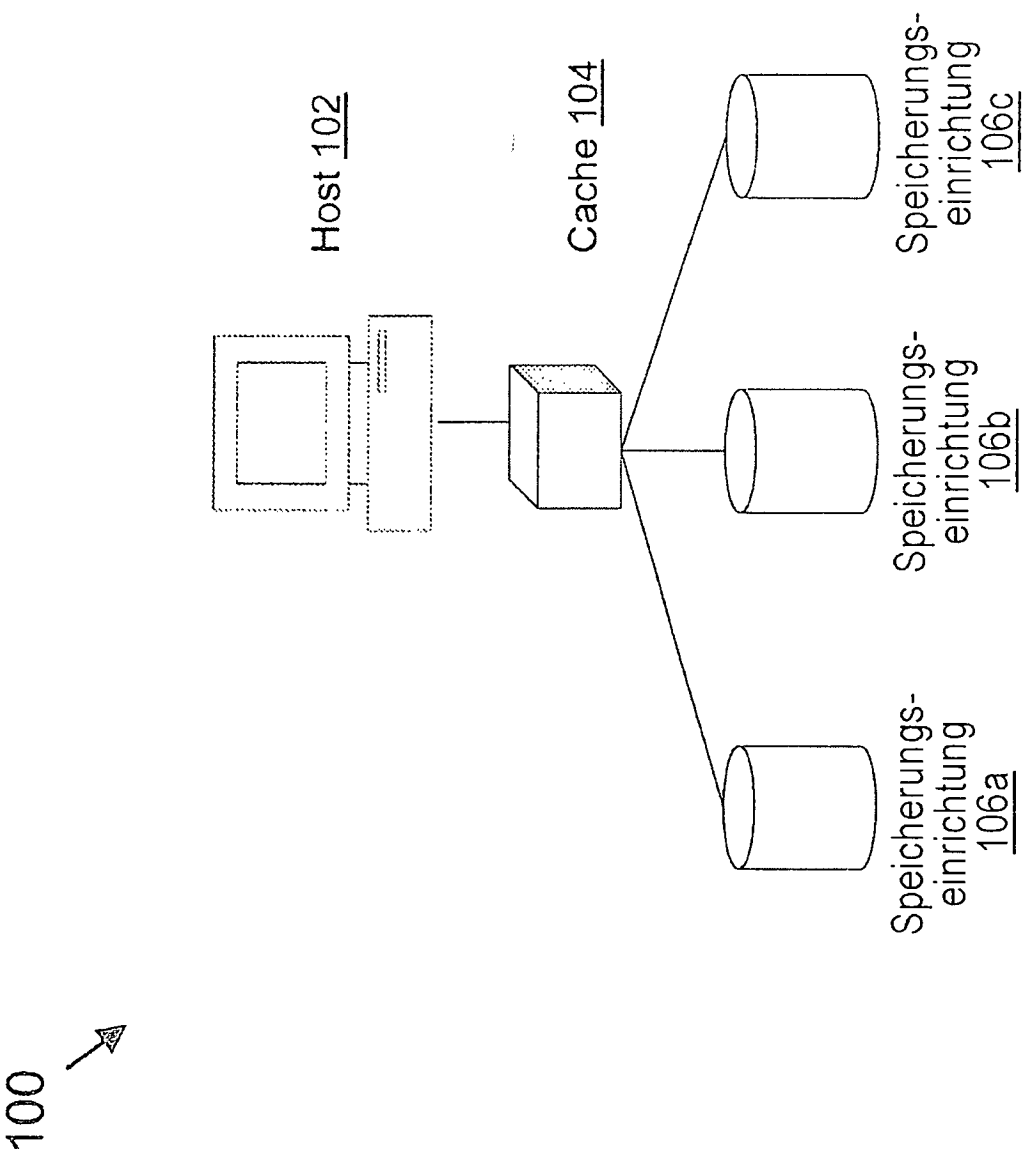


FIG. 1

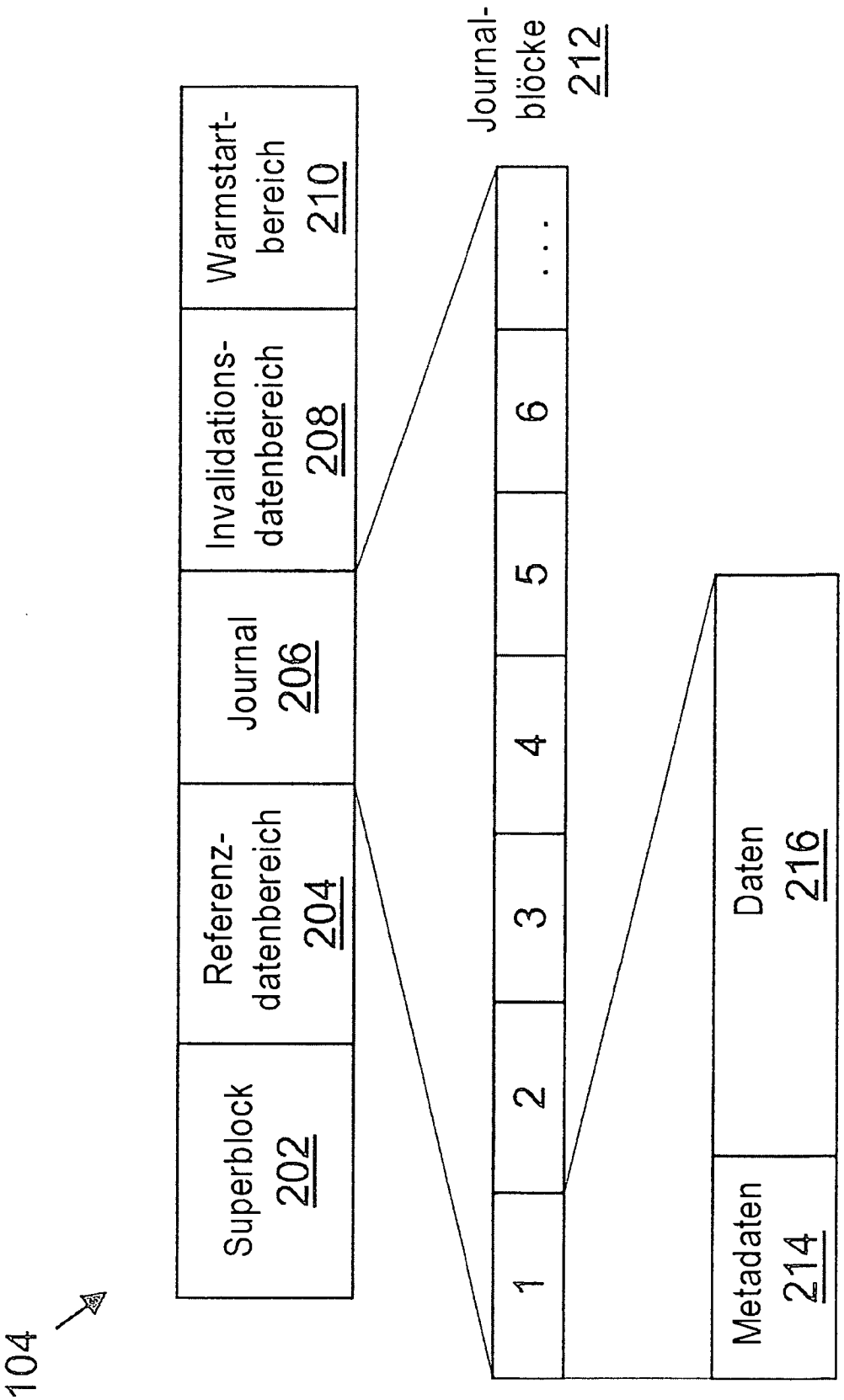

104
Superblock 202
Referenz-datenbereich 204
Journal 206
Invalidations-datenbereich 208
Warmstart-bereich 210
Journal-blöcke 212
1
2
3
4
5
6
. . .
Metadaten 214
Daten 216

FIG. 2A

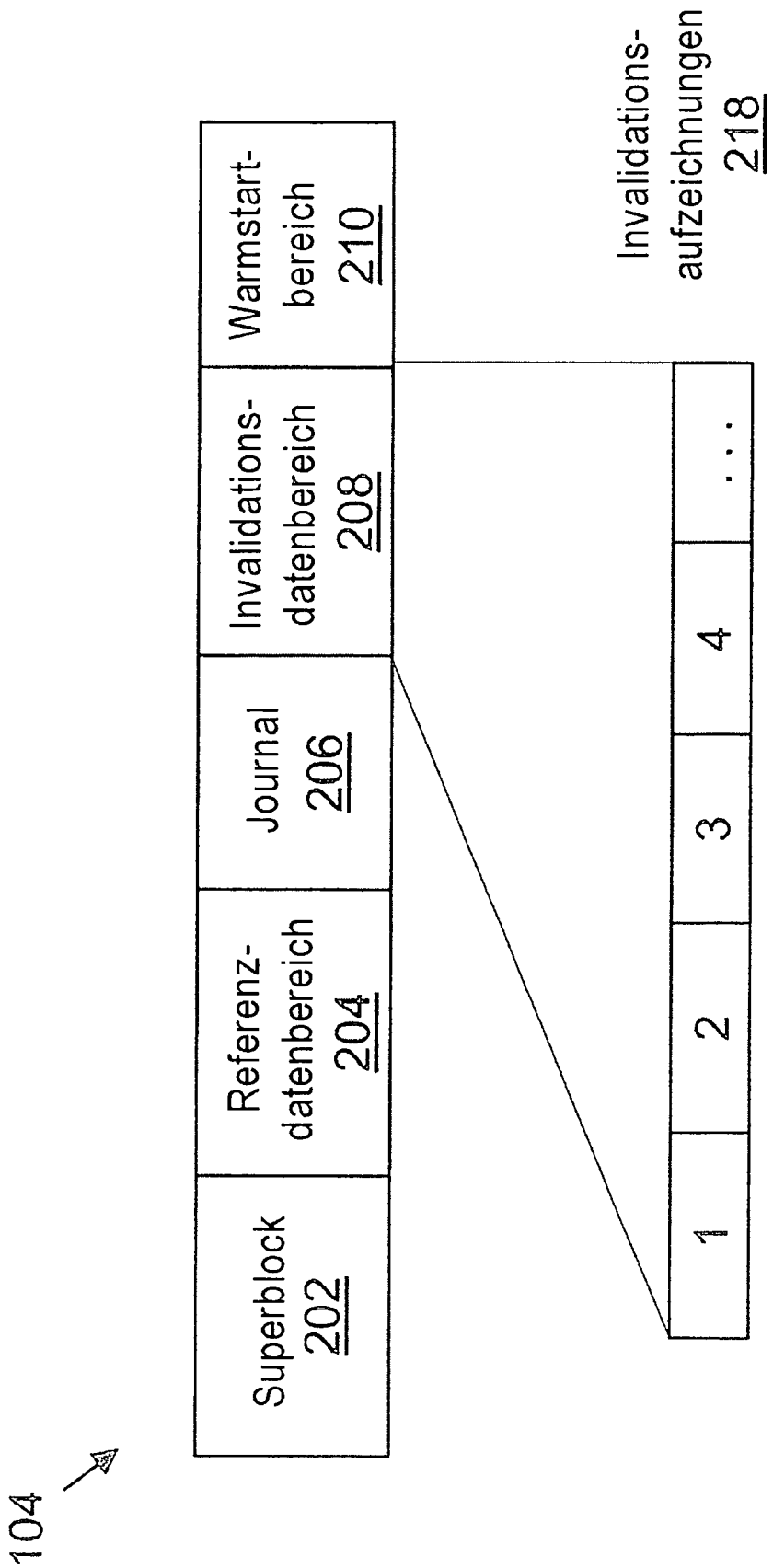

104
Superblock 202
Referenz-datenbereich 204
Journal 206
Invalidations-datenbereich 208
Warmstart-bereich 210
Invalidations-aufzeichnungen 218
1
2
3
4
...

FIG. 2B

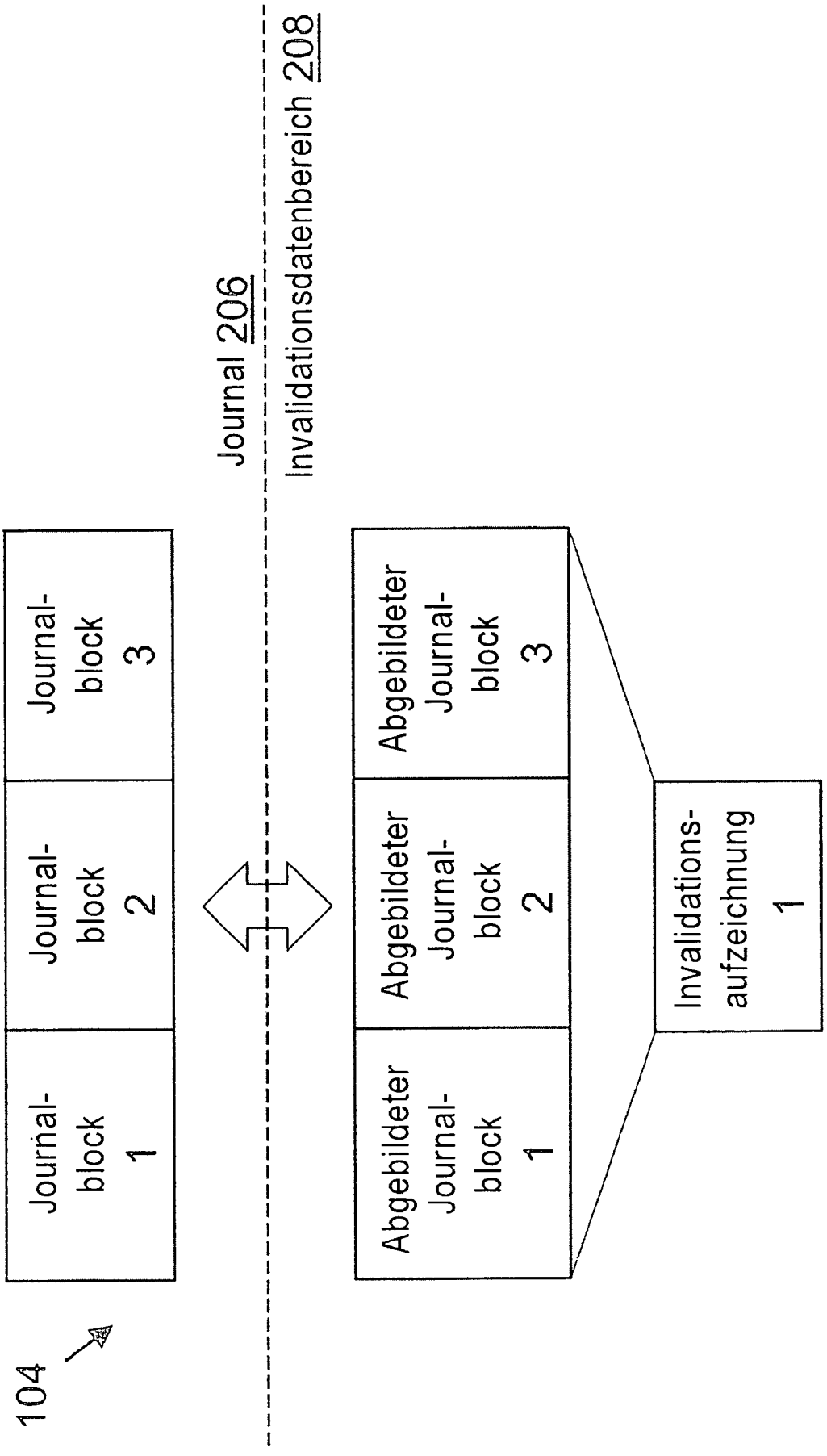

104
Journal-block 1
Journal-block 2
Journal-block 3
Journal 206
Invalidationsdatenbereich 208
Abgebildeter Journal-block 1
Abgebildeter Journal-block 2
Abgebildeter Journal-block 3
Invalidations-aufzeichnung 1

FIG. 3A

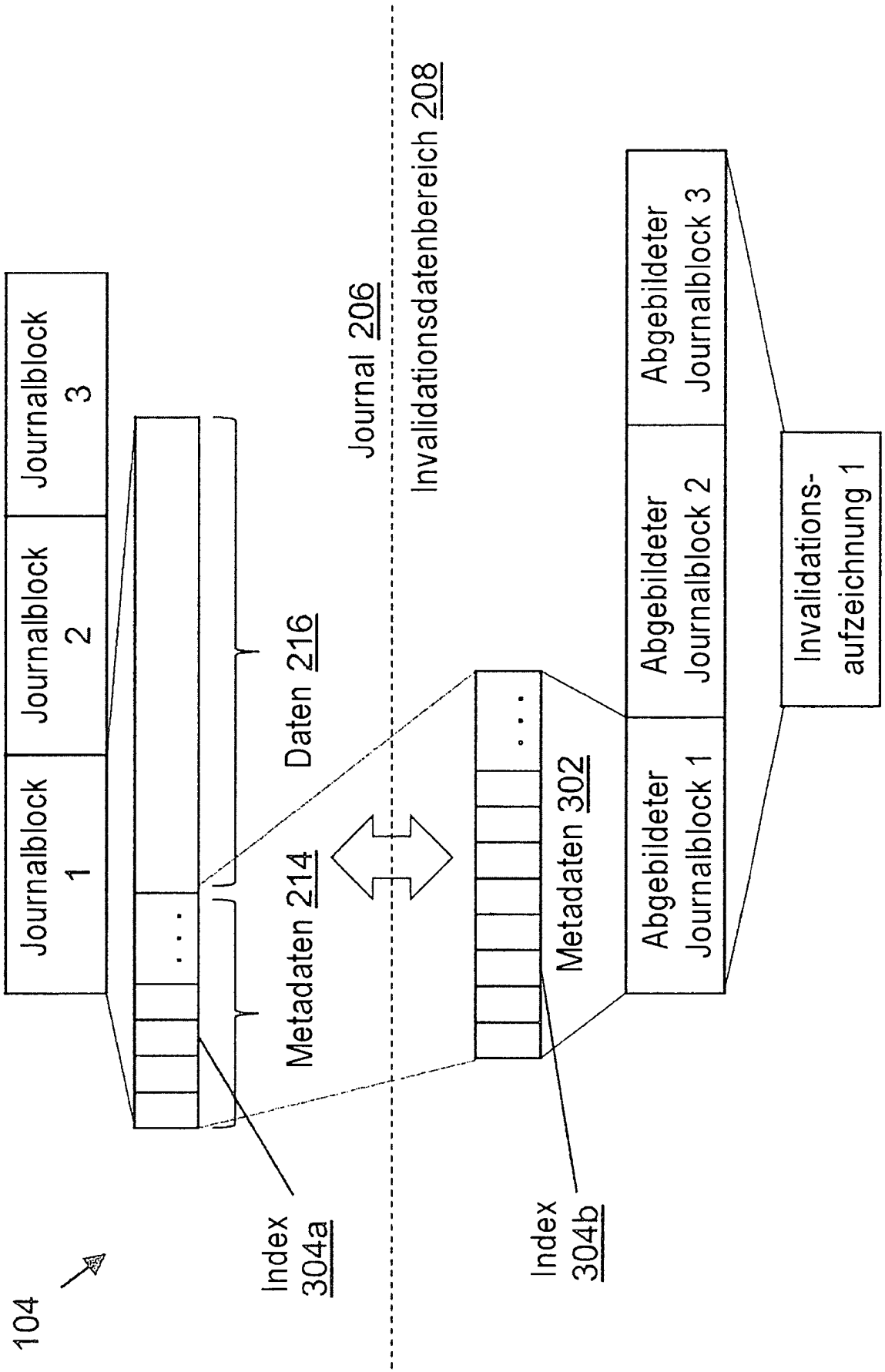

104
Journalblock 1
Journalblock 2
Journalblock 3
Journal 206
Invalidationsdatenbereich 208
Daten 216
Metadaten 214
Index 304a
Metadaten 302
Index 304b
Abgebildeter Journalblock 1
Abgebildeter Journalblock 2
Abgebildeter Journalblock 3
Invalidations-aufzeichnung 1

FIG. 3B

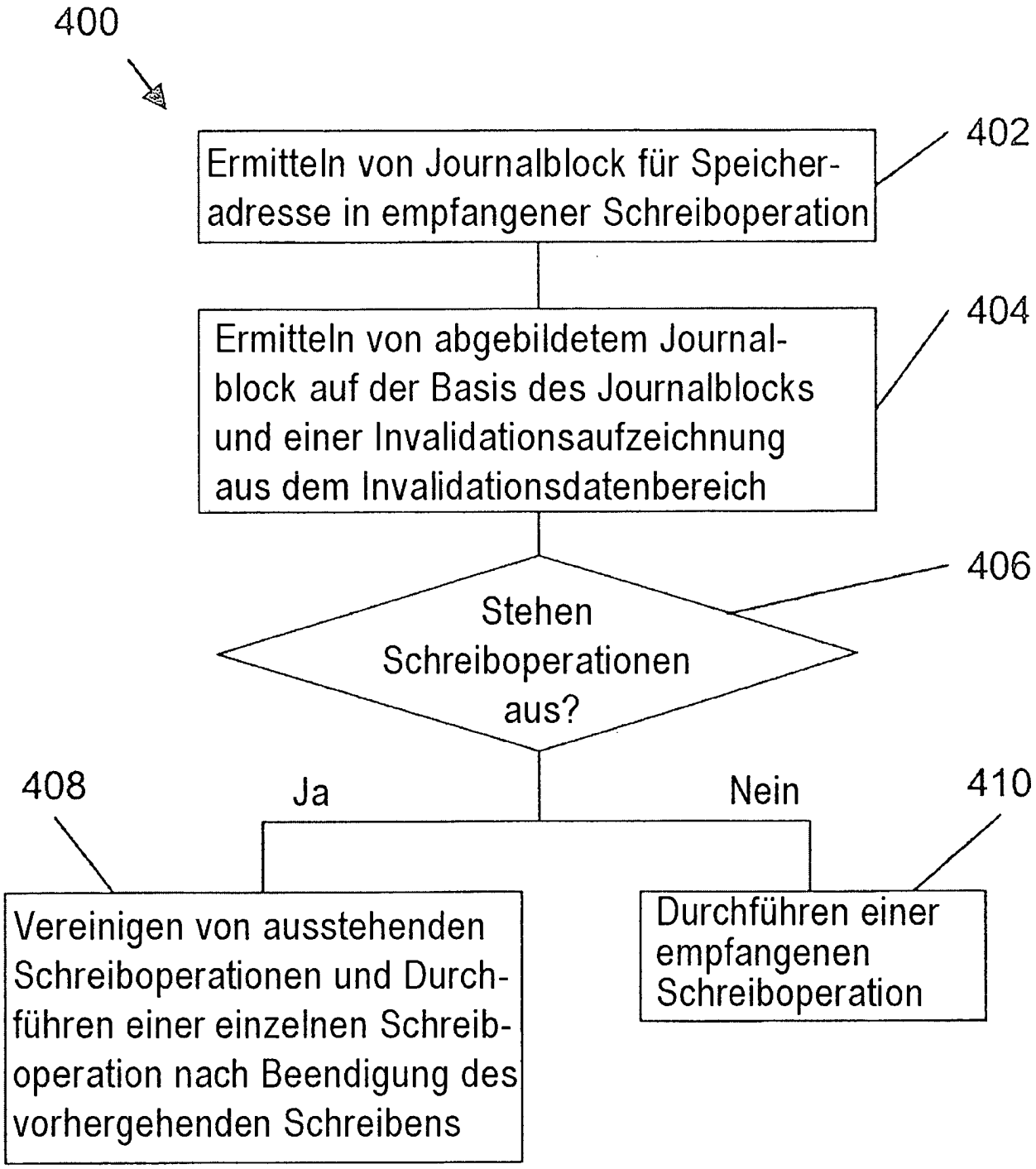

400
Ermitteln von Journalblock für Speicheradresse in empfangener Schreiboperation
402
Ermitteln von abgebildetem Journalblock auf der Basis des Journalblocks und einer Invalidationsaufzeichnung aus dem Invalidationsdatenbereich
404
Stehen Schreiboperationen aus?
406
Ja
Nein
408
410
Vereinigen von ausstehenden Schreiboperationen und Durchführen einer einzelnen Schreiboperation nach Beendigung des vorhergehenden Schreibens
Durchführen einer empfangenen Schreiboperation

FIG. 4

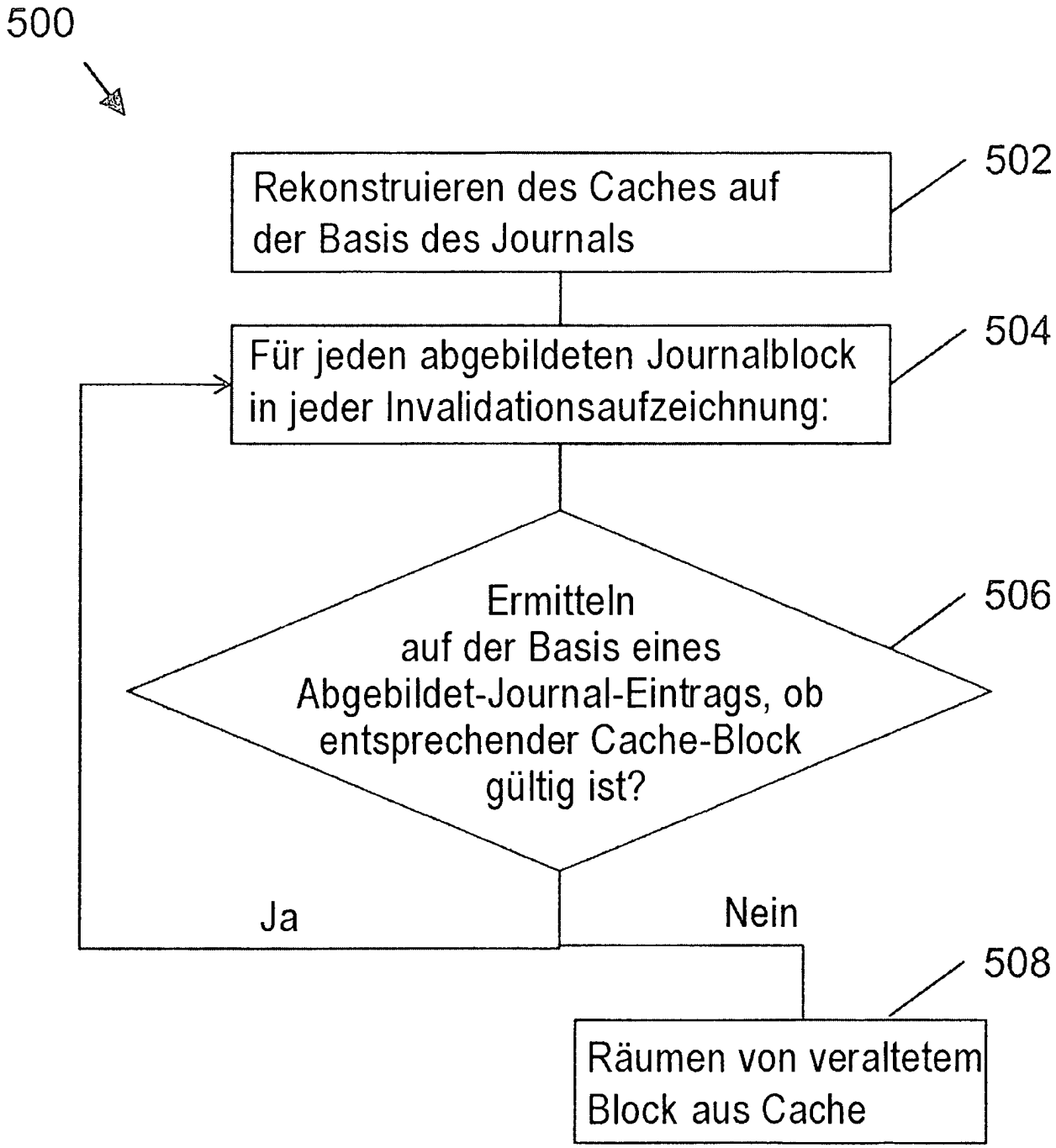


FIG. 5

