(19) Deutsches Patent- und Markenamt

(10) **DE 10 2010 013 263 A1** 2011.01.13

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2010 013 263.2**
(22) Anmeldetag: **29.03.2010**
(43) Offenlegungstag: **13.01.2011**

(51) Int Cl.[8]: ***G06F 12/08*** (2006.01)
G06F 1/32 (2006.01)

(30) Unionspriorität:
12/414,385 30.03.2009 US

(71) Anmelder:
Intel Corporation, Santa Clara, Calif., US

(74) Vertreter:
BOEHMERT & BOEHMERT, 28209 Bremen

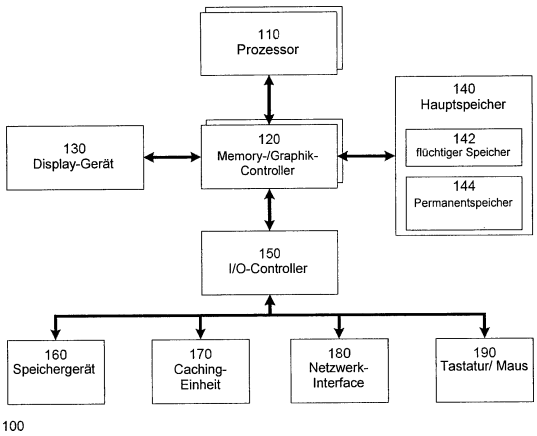
(72) Erfinder:
Trika, Sanjeev N., Hillsboro, Oreg., US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Techniken, um ein energieausfallsicheres Caching ohne atomare Metadaten durchzuführen**

(57) Zusammenfassung: Ein Verfahren und System, um ein energieausfallsicheres Write-back- oder Write-through-Caching von Daten in einem persistenten Speichergerät in eine oder mehrere Cache-Lines einer Caching-Einheit zu ermöglichen. Keine mit irgendeiner der Cache-Lines verbundenen Metadaten werden atomar in die Caching-Einheit geschrieben, wenn die Daten in dem Speichergerät gecacht sind. Somit wird keine besondere Cache-Hardware benötigt, um ein atomares Schreiben von Metadaten während des Cachings von Daten zu ermöglichen.



Beschreibung

GEBIET DER ERFINDUNG

[0001] Diese Erfindung betrifft Caching, und spezieller, aber nicht ausschließlich, energieausfallsicheres Write-back- oder Write-through-Caching in einem nicht flüchtigen Medium.

HINTERGRUND DER BESCHREIBUNG

[0002] Das Speichersubsystem ist eines der langsamsten Subsysteme eines Computersystems, insbesondere wenn das Speichersubsystem ein Speichermedium, wie z. B. ein Festplattenlaufwerk (hard-disk drive, HDD), verwendet. Ein HDD erfordert eine relativ lange Zugriffszeit, da die Schreib-Lese-Köpfe mechanisch zu einem bestimmten Ort auf den Scheiben des HDD bewegt werden müssen, um Daten zu lesen/schreiben.

[0003] Um die Leistung des HDD zu verbessern, kann ein nicht flüchtiger Cache-Speicher verwendet werden, um die Ergebnisse letzter Lesevorgänge von dem HDD und Schreibvorgänge auf das HDD festzuhalten. Dadurch, dass die Daten des HDD gecacht werden, kann die Leistung des Computersystems erhöht werden, und das HDD kann längere Zeit heruntergefahren bleiben, um den Energieverbrauch des Computersystems zu verringern.

[0004] Wenn jedoch die Energiezufuhr an das Computersystem unerwartet abgeschaltet wird, müssen die Daten in dem nicht flüchtigen Cache-Speicher wieder mit dem HDD assoziiert werden, um Datenfehler zu vermeiden. Geeignete Caching-Hardware, die Schreibvorgänge von atomaren Metadaten mit den Schreibvorgängen von Cache-Daten unterstützt, kann verwendet werden, um sicherzustellen, dass diese Wiederherstellung korrekt ausgeführt wird, aber sie erhöht die Kosten des Computersystems.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0005] Die Merkmale und Vorteile von erfindungsgemäßen Ausführungsformen werden aus der folgenden ausführlichen Beschreibung des Gegenstands ersichtlich, bei der:

[0006] Fig. 1 ein System veranschaulicht, um die hierin offenbarten Verfahren in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform zu implementieren;

[0007] Fig. 2 ein Blockdiagramm eines Eingangs-/Ausgangs-(I/O-Input/Output-)Controllers in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0008] Fig. 3 ein Blockdiagramm der Module in einem Betriebssystem in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0009] Fig. 4 eine Konfiguration einer Caching-Einheit in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0010] Fig. 5 ein Ablaufdiagramm eines Write-through-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0011] Fig. 6A ein Ablaufdiagramm eines Write-back-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0012] Fig. 6B ein Ablaufdiagramm eines Write-back-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0013] Fig. 6C ein Ablaufdiagramm eines Write-back-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform veranschaulicht;

[0014] Fig. 7 ein Ablaufdiagramm eines Verfahrens veranschaulicht, um Daten in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform in eine Cache-Line einzufügen, und

[0015] Fig. 8A und Fig. 8B einen Pseudo-Code veranschaulichen, um ein Write-back-Caching-Schema in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform zu implementieren.

AUSFÜHRLICHE BESCHREIBUNG

[0016] Hierin beschriebene erfindungsgemäße Ausführungsformen werden in den beigefügten Figuren in beispielhafter und nicht einschränkender Weise veranschaulicht. Der Einfachheit und Deutlichkeit der Veranschaulichung halber sind in den Figuren veranschaulichte Elemente nicht notwendigerweise maßstabsgetreu. Beispielsweise können die Abmessungen einiger Elemente im Verhältnis zu anderen Elementen zur Verdeutlichung übermäßig groß dargestellt sein. Wo es zweckmäßig erschien, wurden weiter Bezugszeichen in den Figuren wiederholt, um entsprechende oder analoge Elemente zu kennzeichnen. Verweise in der Beschreibung auf „eine erfindungsgemäße Ausführungsform" bedeuten, dass ein bestimmtes Merkmal, eine Struktur oder Charakteristikum, das in Verbindung mit der Ausführungsform beschrieben wird, in zumindest einer erfindungsgemäßen Ausführungsform enthalten ist. Somit bezieht sich das Auftreten des Ausdrucks „bei einer Ausführungsform" an verschiedenen Stellen in der Beschrei-

bung nicht immer zwingend alle auf dieselbe Ausführungsform.

[0017] Erfindungsgemäße Ausführungsformen stellen ein Verfahren und System bereit, um ein energieausfallsicheres Write-back- oder Write-through-Caching von Daten in einem persistenten Speichergerät in eine oder mehr Cache-Lines einer Caching-Einheit zu ermöglichen, die keine atomaren Metadaten erfordert. Keine mit irgendeiner der Cache-Lines verbundenen Metadaten werden atomar in die Caching-Einheit geschrieben, wenn die Daten in dem Speichergerät gecacht sind. Somit wird keine besondere Cache-Hardware benötigt, um ein atomares Schreiben von Metadaten während des Cachens von Daten zu ermöglichen.

[0018] Bei einer erfindungsgemäßen Ausführungsform beinhalten die mit den Cache-Lines verbundenen Metadaten den Ort der Daten auf dem gecachten Speichergerät, wie z. B. die logische Blockadresse (logical block address, LBA) der Daten, die Sequenznummer, den Zustand der Cache-Line, wie z. B. ob die Daten gültig oder ungültig sind, die Verankerungsinformationen oder gecachten LBAs des Speichergeräts und dergleichen, sind aber nicht darauf beschränkt. Das Speichergerät beinhaltet ein Festkörperlaufwerk (solid state drive, SSD), ein HDD, ein RAID-Volumen (Redundant Array of Independent Disks), ein Bandlaufwerk, ein Compact-Disk-(CD-), ein Disketten-, ein universelles serielles Bus-(universal serial bus, USB-)Flash-Speicher-Laufwerk oder jede andere Art Speichermedium für nicht flüchtige oder persistente Computerdaten, ist aber nicht darauf beschränkt. Die Caching-Einheit beinhaltet, ist aber nicht beschränkt auf, ein nicht flüchtiges Medium, ein SSD, einen NAND-Flash-Speicher, einen Phasenwechselspeicher oder jede andere Art Speichermedium für nicht flüchtige oder persistente Computerdaten.

[0019] Fig. 1 veranschaulicht ein System **100**, um die hierin offenbarten Verfahren in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform zu implementieren. Das System **100** beinhaltet, ist aber nicht darauf beschränkt, einen Desktopcomputer, einen Laptop, ein Notebook, ein Netbook, einen Minicomputer (personal digital assistant, PDA), einen Server, eine Workstation, ein Mobiltelefon, ein mobiles EDV-Gerät, ein Internet-Gerät oder jede andere Art EDV-Gerät. Bei einer weiteren Ausführungsform kann das System **100**, das verwendet wird, um die hierin offenbarten Verfahren zu implementieren, ein Ein-Chip-System (system an a chip, SOC) sein.

[0020] Das System **100** beinhaltet einen Memory-/Graphik-Controller **120** und einen I/O-Controller **150**. Der Memory-/Graphik-Controller **120** stellt typischerweise Speicher- und I/O-Managementfunktionen bereit, sowie eine Vielzahl von Universal- und/oder Spezialregistern, Zeitgebern etc., auf die vom Prozessor **110** zugegriffen werden kann oder die von ihm verwendet werden können. Der Prozessor **110** kann unter Verwendung eines oder mehrerer Prozessoren oder unter Verwendung von Mehrkemprozessoren implementiert sein. Der I/O-Controller **150** ermöglicht ein energieausfallsicheres Write-back- oder Write-through-Caching von Daten im Speichergerät **160** in eine oder mehrere Cache-Lines der Caching-Einheit **170** oder des Permanentspeichers **144** in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform.

[0021] Der Memory-/Graphik-Controller **120** führt Funktionen aus, die es dem Prozessor **110** ermöglichen, auf einen Hauptspeicher **140**, der einen flüchtigen Speicher **142** und/oder einen Permanentspeicher **144** beinhaltet, zuzugreifen und damit zu kommunizieren. Bei einer weiteren erfindungsgemäßen Ausführungsform ist ein weiterer flüchtiger Speicher **142** (nicht in Fig. 1 gezeigt) in das Speichergerät **160** eingebaut, um die Daten des Speichergeräts **160** zu cachen. Der Memory-/Graphik-Controller **120** kann, anstatt des I/O-Controllers **150**, ein energieausfallsicheres Write-back- oder Write-through-Caching von Daten in dem Speichergerät **160** in die eine oder mehrere Cache-Lines der Caching-Einheit **170** in Übereinstimmung mit einer weiteren erfindungsgemäßen Ausführungsform ermöglichen.

[0022] Der flüchtige Speicher **142** beinhaltet, ist aber nicht beschränkt auf, Synchronous Dynamic Random Access Memory (SDRAM), Dynamic Radom Access Memory (DRAM), RAMBUS DRAM (RDRAM) und/oder jede andere Art Speichergerät mit wahlfreiem Zugriff (random access). Der Permanentspeicher **144** beinhaltet, ist aber nicht beschränkt auf, NAND-Flash-Speicher, Festspeicher (read only memory, ROM), elektrisch löschbarer programmierbarer ROM (electrically erasable programmable ROM, EEPROM) und/oder jede andere gewünschte Art Speichergerät. Der Hauptspeicher **140** speichert Informationen und Befehle, die von dem Prozessor **110** ausgeführt werden sollen. Der Hauptspeicher **140** kann ebenfalls temporäre Variablen oder andere Zwischeninformationen speichern, während der Prozessor **110** Befehle ausführt. Bei einer weiteren erfindungsgemäßen Ausführungsform ist der Memory-/Graphik-Controller **120** Teil des Prozessors **110**.

[0023] Der Memory-/Graphik-Controller **120** ist mit einem Display-Gerät **130** verbunden, das Flüssigkristallbildschirme (liquid crystal displays, LCDs), Röhrenmonitor-(cathode ray tube, CRT-)Displays oder jede andere Art von visuellem Display-Gerät beinhaltet, aber nicht darauf beschränkt ist. Der I/O-Controller **150** ist mit einem Speichergerät(en) **160**, einer Caching-Einheit(en) **170**, einem Netzwerk-Interface **180** und einer Tastatur/Maus **190** gekoppelt, aber

nicht darauf beschränkt. Insbesondere führt der I/O-Controller **150** Funktionen aus, die es dem Prozessor **110** ermöglichen, mit dem Speichergerät **160**, der Caching-Einheit **170**, dem Netzwerk-Interface **180** und der Tastatur/Maus **190** zu kommunizieren. Bei einer Ausführungsform könnte die Caching-Einheit **170** Teil des Speichergeräts **160** sein.

[0024] Das Netzwerk-Interface **180** ist unter Verwendung jeder Art wohl bekannten Netzwerk-Interface-Standards implementiert und beinhaltet, ist aber nicht darauf beschränkt, eine Ethernet-Schnittstelle, eine USB-Schnittstelle, eine Peripheral-Component-Interconnect-(PCI-)Express-Schnittstelle, eine Drahtlosschnittstelle und/oder jede andere geeignete Art von Schnittstelle. Die Drahtlosschnittstelle arbeitet in Übereinstimmung mit, ist aber nicht darauf beschränkt, der Drahtlosstandardfamilie Institute of Electrical and Electronics Engineers (IEEE) 802.11, Home Plug AV (HPAV), Ultra-Breitband (utra wide band, UWB), Bluetooth, WiMax oder jeder anderen Art von Drahtlos-Kommunikationsprotokoll.

[0025] Bei einer erfindungsgemäßen Ausführungsform ist/sind der/die in Fig. 1 gezeigte(n) Bus(se) ein von allen damit verbundenen Komponenten gemeinsam benutzter Kommunikationslink. Bei einer weiteren erfindungsgemäßen Ausführungsform ist/sind der/die in Fig. 1 gezeigte(n) Bus(se) ein Punkt-zu-Punkt-Kommunikationslink zwischen Komponentenpaaren, die miteinander verbunden sind. Während die in Fig. 1 gezeigten Komponenten als getrennte Blöcke innerhalb des Systems **100** dargestellt sind, können die von einigen dieser Blöcke ausgeführten Funktionen innerhalb einer einzigen Halbleiterschaltung integriert sein oder können unter Verwendung von zwei oder mehreren getrennten integrierten Schaltungen implementiert sein. Obwohl beispielsweise der Memory-/Graphik-Controller **120** und der I/O-Controller **150** als getrennte Blöcke dargestellt sind, ist es für einen Fachmann sofort selbstverständlich, dass der Memory-/Graphik-Controller **120** und der I/O-Controller **150** innerhalb einer einzigen Halbleiterschaltung integriert sein können.

[0026] Fig. 2 veranschaulicht ein Blockdiagramm **200** eines I/O-Controllers **150** in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. Der I/O-Controller **150** verfügt über einen Wiederherstellungs-Controller **212** und einen Laufzeit-Controller **214**. Bei einer erfindungsgemäßen Ausführungsform weist der Laufzeit-Controller **214** eine auf Heuristiken basierende Caching-Richtlinie auf, um zu bestimmen, ob die Daten des Speichergeräts **160** gecacht oder aus der Caching-Einheit **170** entfernt werden sollen. Die Heuristiken beinhalten, sind aber nicht beschränkt auf, zuletzt aufgerufene LBAs, Verankerungsinformationen der LBAs und dergleichen. Der Laufzeit-Controller **214** führt ebenfalls Caching-Mechanismen aus, wie z. B. das Erkennen von Cache-Hits oder Cache Misses, und das Queuing von Befehlen zum Cachen oder Entfernen bei einer erfindungsgemäßen Ausführungsform.

[0027] Der Laufzeit-Controller **214** verwendet die volle Datenkapazität der Caching-Einheit **170**, um die Daten des Speichergeräts **160** bei einer erfindungsgemäßen Ausführungsform zu cachen. Bei einer weiteren erfindungsgemäßen Ausführungsform verwendet der Laufzeit-Controller **214** einen Teil der vollen Datenkapazität der Caching-Einheit **170**, um die Daten des Speichergeräts **160** zu cachen. Bei einer erfindungsgemäßen Ausführungsform beispielsweise verwendet der Laufzeit-Controller **214** die Hälfte der vollen Datenkapazität der Caching-Einheit **170**, um die Daten des Speichergeräts **160** zu cachen, und verwendet die andere Hälfte der vollen Datenkapazität der Caching-Einheit **170** als ein Speichermedium.

[0028] Der Wiederherstellungs-Controller **212** und der Laufzeit-Controller **214** ermöglichen bei einer erfindungsgemäßen Ausführungsform ein energieausfallsicheres Write-back- oder Write-through-Caching der Daten in dem Speichergerät **160** in die Caching-Einheit **170**. Für einen Fachmann ist es sofort selbstverständlich, dass andere Caching-Schemata ebenfalls verwendet werden können, ohne die erfindungsgemäßen Arbeiten zu beeinträchtigen. Im Falle eines Systemfehlers **100** bewahren bei einer erfindungsgemäßen Ausführungsform der Wiederherstellungs-Controller **212** und der Laufzeit-Controller **214** die Integrität oder Kohärenz der Daten in dem Speichergerät **160** und der gecachten Daten in der Caching-Einheit **170**. Das Fehlerereignis des Systems **100** beinhaltet, ist aber nicht beschränkt auf, Fehler durch Energieverlust, Fehler durch Absturz des Betriebssystems (operating system, OS), unsachgemäße Beendigung des Systems **100** und andere Ereignisse, die nicht innerhalb der normalen Betriebsbedingungen des Systems **100** liegen.

[0029] Der Wiederherstellungs-Controller **212** stellt den Cache-Zustand der Cache-Lines in der Caching-Einheit **170** wieder her, nachdem ein Fehlerereignis bei einer erfindungsgemäßen Ausführungsform aufgetreten ist. Bei weiteren erfindungsgemäßen Ausführungsformen verarbeitet der Wiederherstellungs-Controller **212** weitere Ereignisse, einschließlich, aber nicht darauf beschränkt, Abstandserkennung und Verarbeitung, Verarbeitung aller I/O-Daten vor der Initialisierung des Laufzeit-Controllers **214** und dergleichen. Obwohl der Wiederherstellungs-Controller **212** und der Laufzeit-Controller **214** als Teil des I/O-Controllers **150** in Fig. 2 dargestellt sind, soll dies als nicht einschränkend verstanden werden. Der Wiederherstellungs-Controller **212** und der Laufzeit-Controller **214** können gemeinsam in demselben Hardware- oder Softwaremodul implementiert sein oder sie können getrennt in verschiedenen Hardware- oder Softwaremodulen implementiert

sein.

[0030] Bei einer erfindungsgemäßen Ausführungsform sind der Wiederherstellungs-Controller **212** und der Laufzeit-Controller **214** Teil des Memory-/Graphik-Controllers **120**. Der Wiederherstellungs-Controller **212** und der Laufzeit-Controller **214** können ebenfalls als ein einziger Controller bei einer weiteren erfindungsgemäßen Ausführungsform zusammengefasst sein. Für einen Fachmann ist es sofort selbstverständlich, dass verschiedene Konfigurationen des Wiederherstellungs-Controllers **212** und des Laufzeit-Controllers **214** implementiert werden können, ohne die erfindungsgemäßen Arbeiten zu beeinträchtigen. Bei einer erfindungsgemäßen Ausführungsform beispielsweise ist der Wiederherstellungs-Controller **212** als eine in einem Options-ROM des Systems **100** gespeicherte Firmware implementiert und der Laufzeit-Controller **214** ist in einem Blockspeichertreiber eines auf dem System **100** ablaufenden OS implementiert.

[0031] Fig. 3 veranschaulicht ein Blockdiagramm **300** der Module in einem OS in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. Das OS hat eine Anwendungsschicht **310** und ein Dateisystem **320**. Die Anwendungsschicht **310** ist in der Lage, auf Dateien zuzugreifen, die von dem Dateisystem **320** organisiert sind. Das OS verfügt ebenfalls über einen Speichertreiberstapel **330** und einen Blocktreiber **340**. Der Blocktreiber **340** verfügt über einen Laufzeit-/Wiederherstellungs-Controller **344** in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. Der Blocktreiber **340** kann den Laufzeit-Controller, den Wiederherstellungs-Controller oder sowohl den Laufzeit- als auch den Wiederherstellungs-Controller beinhalten.

[0032] Der Laufzeit-/Wiederherstellungs-Controller **344** ist mit dem Speichergerät **160** und der Caching-Einheit **170** gekoppelt, und er cacht die Daten in dem Speichergerät **160** in die Caching-Einheit **170**. Keine Zustandsinformationen oder mit irgendeiner der Cache-Lines der Caching-Einheit **170** verbundenen Metadaten werden atomar in der Caching-Einheit **170** während des Cachens der Daten in dem Speichergerät **160** gespeichert. Bei einer erfindungsgemäßen Ausführungsform verwendet das OS ein Write-back Caching-Schema, bei dem jegliche Daten, die auf das Speichergerät **160** geschrieben werden sollen, zuerst auf die Caching-Einheit **170** geschrieben werden. Das OS schreibt die Daten nicht auf das Speichergerät **160**, sofort nach dem Schreiben auf die Caching-Einheit **170**, sondern wartet eine angemessene Zeit, um die Daten auf das Speichergerät zu schreiben. Somit ist der Datenzugriff des Speichergeräts **160** minimiert, und das OS braucht nicht darauf zu warten, dass die Daten auf das Speichergerät **160** geschrieben werden, bevor weitere Befehle ausgeführt werden. Das Write-back-Caching-Schema ist vorteilhaft, um das Speichersubsystem des Systems **100** zu beschleunigen, da die Datenzugriffsgeschwindigkeit der Caching-Einheit **170** schneller ist, als die Datenzugriffsgeschwindigkeit des Speichergeräts **160**.

[0033] Es kann sein, dass die Daten in dem Speichergerät **160** nicht mit den gecachten Daten in der Caching-Einheit **170** synchron sind, wenn ein Write-back-Caching-Schema verwendet wird. Bei einer erfindungsgemäßen Ausführungsform synchronisiert der Laufzeit-/Wiederherstellungs-Controller **344** die gecachten Daten in der Caching-Einheit **170** mit den Daten im Speichergerät **160**, wenn der Nutzungsgrad des Prozessors **110**, Speichergeräts **160** oder Caching-Einheit **170** des Systems **100** unter dem verwendeten liegt. Bei einer erfindungsgemäßen Ausführungsform beispielsweise bestimmt der Laufzeit-/Wiederherstellungs-Controller **344**, dass der Nutzungsgrad des Prozessors **110** bei dem System **100** unterhalb eines Grenzwertes liegt und synchronisiert die gecachten Daten in der Caching-Einheit **170**, die nicht mit den Daten in dem Speichergerät **160** synchronisiert worden sind. Für einen Fachmann ist es sofort selbstverständlich, dass weitere Schemata oder Richtlinien verwendet werden können, um eine Hintergrundsynchronisation der Daten in der Caching-Einheit **170** durchzuführen, ohne die erfindungsgemäßen Arbeiten zu beeinträchtigen.

[0034] Das OS kann einen periodischen Flush-Befehl an den I/O-Controller **150** ausgeben, um sicherzustellen, dass alle vorher geschriebenen Daten nicht flüchtig sind. Bei einer erfindungsgemäßen Ausführungsform garantiert der I/O-Controller **150**, dass sowohl die Aktualisierungen der Daten als auch der Metadaten in dem Speichergerät **160** und der Caching-Einheit **170** nicht flüchtig sind, wenn ein Flush-Befehl ausgeführt wird, und dass alle vorher geschriebenen Daten wiederhergestellt werden können, auch wenn eine unsachgemäße Beendigung, wie z. B. ein Stromausfall des Systems **100** auftritt.

[0035] Bei einer weiteren erfindungsgemäßen Ausführungsform verwendet das OS ein Write-through-Caching-Schema, bei dem die Daten in dem Speichergerät **160** und die gecachten Daten in der Caching-Einheit **170** immer synchron sind. Wenn das OS eine Schreiboperation durchführt, werden sowohl die Caching-Einheit **170** als auch das Speichergerät **160** mit denselben Daten beschrieben.

[0036] Erfindungsgemäße Ausführungsformen ermöglichen, dass die Entwicklungskosten des Systems **100** verringert werden, da keine besondere Caching-Hardware erforderlich ist, um ein energieausfallsicheres Write-through- und Write-back-Caching zu ermöglichen. Bei einer erfindungsgemäßen Ausführungsform beispielsweise wird ein relativ kleines SSD verwendet, um ein oder mehrere große Fest-

plattenlaufwerke, ohne besondere Caching-Hardware zu benötigen, zu cachen.

[0037] Fig. 4 veranschaulicht eine Konfiguration **400** einer Caching-Einheit **170** in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. Die Konfiguration **400** der Caching-Einheit **170** zeigt ein logisches Segment gepackter Metadaten **401** und ein weiteres logisches Segment an Cache-Lines **402**. Die Blockbreite **405** der Caching-Einheit **170** zeigt die Datenbitbreite der Caching-Einheit **170**. Bei einer weiteren erfindungsgemäßen Ausführungsform kann die Konfiguration **400** der Caching-Einheit **170** ebenfalls weitere logische Segmente (nicht gezeigt in Fig. 4) beinhalten, die für andere Zwecke verwendet werden, wie z. B. Datensicherung oder Datenindizierung.

[0038] Beispielsweise ist das logische Segment von Cache-Lines **402** mit acht Cache-Lines (Cache-Lines 0 bis 7) dargestellt, die verwendet werden, um die Daten des Speichergeräts **160** zu cachen. Das logische Segment von Cache-Lines **402** beinhaltet keinerlei mit irgendeiner der Cache-Lines **402** verbundenen Metadaten. Für einen Fachmann ist es sofort selbstverständlich, dass die Caching-Einheit **170** mehr als acht Cache-Lines haben kann, um die Daten des Speichergeräts **160** zu cachen. Bei einer erfindungsgemäßen Ausführungsform speichert jede Cache-Line der Caching-Einheit **170** angrenzende Daten des Speichergeräts **160**. Bei einer weiteren erfindungsgemäßen Ausführungsform speichert jede Cache-Line von Caching-Einheit **170** angrenzende Daten des Speichergeräts **160** nicht. Die Blockbreite **405** ist nicht auf eine bestimmte Bitbreite beschränkt. Bei einer erfindungsgemäßen Ausführungsform entspricht die Blockbreite **405** der Busbreite des Kommunikationslinks zwischen der Caching-Einheit und dem Laufzeit-/Wiederherstellungs-Controller **344**. Wenn bei einer erfindungsgemäßen Ausführungsform beispielsweise die Busbreite des Kommunikationslinks zwischen der Caching-Einheit und dem Laufzeit-/Wiederherstellungs-Controller **344** 64 Bit beträgt, kann die Blockbreite **405** bei einer Bitbreite gesetzt werden, die einem Vielfachen von 64 Bit entspricht. Bei einer weiteren erfindungsgemäßen Ausführungsform wird die Blockbreite **405** gesetzt, um ein Vielfaches der LBAs des Speichergeräts **160** zu speichern. Beispielsweise wird jede Cache-Line der Caching-Einheit auf eine Blockbreite **405** gesetzt, die vier LBAs des Speichergeräts **160** speichern kann.

[0039] Bei dem logischen Segment gepackter Metadaten **401** sind Metadaten 0 bis 7 in einem gepackten Format gespeichert, sodass mehrere Metadaten, jede mit einer anderen Cache-Line verbunden, bei einer erfindungsgemäßen Ausführungsform nebeneinander gespeichert werden. Beispielsweise werden Metadaten 0 **410** mit der Cache-Line 0 **450** verbunden, Metadaten 1 **411** werden mit Cache-Line 1 **451** verbunden und so weiter. Bei einer erfindungsgemäßen Ausführungsform weisen die gepackten Metadaten **401** eine Integritätssignatur für jeden Block Metadaten auf. Die Metadaten 0 bis 3 **410**, **411**, **412** und **413** weisen eine Integritätssignatur 1 **430** auf, und die Metadaten 4 bis 7 **414**, **415**, **416** und **417** weisen eine Integritätssignatur 2 **440** auf. Die Integritätssignaturen **430** und **440** schützen vor beschädigten Datenstrukturen aufgrund einer unerwarteten Beendigung von System **100** oder aufgrund eines Fehlerereignisses. Das logische Segment gepackter Metadaten **401** ist angrenzend in der Caching-Einheit **170** angeordnet, um einen schnelleren Zugriff der gepackten Metadaten **401** bei einer erfindungsgemäßen Ausführungsform zu ermöglichen. Bei einer weiteren erfindungsgemäßen Ausführungsform ist das logische Segment gepackter Metadaten **401** nicht angrenzend in der Caching-Einheit **170** angeordnet. Bei noch einer weiteren Ausführungsform sind die Integritätssignaturen **430** und **440** nicht in dem logischen Segment gepackter Metadaten **401** gespeichert.

[0040] Um ein Write-back- oder Write-through-Caching in der Caching-Einheit **170** zu erleichtern, verwaltet das OS bei einer erfindungsgemäßen Ausführungsform Informationen der Cache-Lines in dem flüchtigen Speicher **142**. Die Informationen der Cache-Lines beinhalten, sind aber nicht beschränkt auf, eine Liste von Cache-Lines, die unbenutzt sind oder die keinerlei Daten des Speichergeräts **160** halten, eine Cache-Tabelle, die Linkinformationen zwischen den Daten oder LBA in dem Speichergerät **160** und den Cache-Lines in der Caching-Einheit **170** aufweist, die die Daten oder LBA speichert, Metadaten aller Cache-Lines in der Caching-Einheit **170**, die in gepacktem Format oder einem anderen Format gespeichert werden können, eine Liste von Cache-Lines, für die ihre entsprechenden Metadaten in dem flüchtigen Speicher **142** noch auf die Metadaten in der Caching-Einheit **170** geschrieben werden müssen, und dergleichen. Bei einer erfindungsgemäßen Ausführungsform verwaltet das OS eine Kopie des logischen Segments gepackter Metadaten **401** der Caching-Einheit **170** in dem flüchtigen Speicher **142**, um ein Write-back- oder Write-through-Caching in der Caching-Einheit **170** zu erleichtern. Bei einer erfindungsgemäßen Ausführungsform kann die Cache-Tabelle als eine Hash-Tabelle, ein Baum oder jede andere Datenstruktur zum Suchen implementiert sein.

[0041] Fig. 5 veranschaulicht ein Ablaufdiagramm **500** eines Write-through-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. In Schritt **510** prüft der Laufzeit-Controller, ob ein Fehlerereignis aufgetreten ist. Bei einer erfindungsgemäßen Ausführungsform prüft der Laufzeit-Controller ein Register oder Flag, das anzeigt, ob ein Fehlerereignis aufgetreten ist. Bei einer erfindungsgemäßen Ausführungsform prüft Schritt **510**,

ob das System **100** unsachgemäß heruntergefahren wurde. Bei einer weiteren erfindungsgemäßen Ausführungsform prüft Schritt **510**, ob das OS abgestürzt ist oder nicht funktioniert. Wenn es ein Fehlerereignis gibt, setzt der Laufzeit-Controller die Caching-Einheit **170** in Schritt **512** zurück.

[0042] Der Ablauf geht zurück zu Schritt **510**, nachdem die Caching-Einheit **170** in Schritt **512** zurückgesetzt wurde. Bei einer erfindungsgemäßen Ausführungsform setzt der Laufzeit-Controller die Caching-Einheit **170** zurück, indem alle Cache-Lines der Caching-Einheit **170** zu einer Liste freier oder unbenutzter Cache-Lines hinzugefügt werden. Die Liste freier Cache-Lines zeigt dem Laufzeit-Controller an, dass die Cache-Lines in der Liste verfügbar sind, um die Daten des Speichergeräts **160** zu cachen. Bei einer weiteren erfindungsgemäßen Ausführungsform setzt der Laufzeit-Controller die Caching-Einheit **170** zurück, indem er alle Cache-Lines der Caching-Einheit **170** als unbenutzt auszeichnet oder markiert.

[0043] Wenn es kein Fehlerereignis gibt, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, das System **100** in Schritt **520** sachgemäß herunterzufahren. Ein sachgemäßes Herunterfahren oder Beenden des Systems **100** betrifft ein Ereignis, bei dem das OS einen Befehl an das System **100** ausgibt, einschließlich, aber nicht beschränkt auf, einen Neustartbefehl, einen Befehl zur Beendigung, einen Hibernate-Befehl, einen Standby-Befehl oder jeden Befehl, der das System **100** herunterfährt. Wenn eine Anfrage zum sachgemäßen Herunterfahren des Systems **100** vorliegt, kopiert der Laufzeit-Controller die gepackten Metadaten, die mit all den Cache-Lines der Caching-Einheit **170** verbunden sind, in Schritt **522** von dem flüchtigen Speicher **142** in die Caching-Einheit **170**. Bei einer erfindungsgemäßen Ausführungsform kopiert der Laufzeit-Controller die gepackten Metadaten, die mit all den Cache-Lines der Caching-Einheit **170** verbunden sind, von dem flüchtigen Speicher **142** in das logische Segment gepackter Metadaten **401**. Im optionalen Schritt **524** kopiert der Laufzeit-Controller die Cache-Tabelle von dem flüchtigen Speicher **142** in die Caching-Einheit **170** und der Ablauf **500** geht zu Schritt **510** zurück.

[0044] Wenn keine Anfrage zum sachgemäßen Herunterfahren des Systems **100** vorliegt, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, Daten in Schritt **530** in die Cache-Line(s) der Caching-Einheit **170** zu aktualisieren oder einzufügen. Wenn bei einer erfindungsgemäßen Ausführungsform beispielsweise das OS Daten auf einen bestimmten Adressort in dem Speichergerät **160** schreiben möchte, prüft der Laufzeit-Controller die Cache-Tabelle, ob die Daten an dem bestimmten Adressort in dem Speichergerät **160** in die Caching-Einheit **170** gecacht sind. Wenn es einen Cache-Hit gibt, d. h. die Daten an dem bestimmten Adressort sind in der Caching-Einheit **170** gecacht, empfängt der Laufzeit-Controller eine Anfrage, um die übereinstimmende(n) Cache-Line(s), die die Daten des bestimmten Adressortes speichert/speichern, zu aktualisieren. Wenn es einen Cache-Miss gibt, d. h. die Daten an dem bestimmten Adressort sind nicht in der Caching-Einheit **170** gecacht, empfängt der Laufzeit-Controller eine Anfrage, um die Daten des bestimmten Adressortes in die Cache-Line(s) der Caching-Einheit **170** einzufügen.

[0045] Wenn eine Anfrage vorliegt, Daten in die Cache-Line(s) der Caching-Einheit **170** zu aktualisieren oder einzufügen, aktualisiert der Laufzeit-Controller in Schritt **532** die gepackten Metadaten oder Zustandsinformationen, die mit der/den Cache-Line(s) verbunden sind, basierend auf den neuen zu schreibenden Daten, in dem flüchtigen Speicher **142**. In Schritt **534** aktualisiert der Laufzeit-Controller die Cache-Line(s) und das Speichergerät **160** mit den neuen Daten. Die Daten in der Caching-Einheit **170** und dem Speichergerät **160** werden synchronisiert, wenn Schritt **534** ausgeführt ist.

[0046] Wenn keine Anfrage vorliegt, Daten in der/den Cache-Line(s) der Caching-Einheit **170** zu aktualisieren oder einzufügen, prüft der Wiederherstellungs-Controller, ob irgendeine Benachrichtigung zum Hochfahren des Systems **100** in Schritt **540** vorliegt. Wenn ja, speichert der Wiederherstellungs-Controller die gepackten Metadaten in der Caching-Einheit **170** in Schritt **542** erneut in den flüchtigen Speicher **142** oder kopiert sie dorthin. Im optionalen Schritt **544** speichert der Wiederherstellungs-Controller die Cache-Tabelle in der Caching-Einheit **170** erneut in den flüchtigen Speicher **142** oder kopiert sie dorthin, wenn die Cache-Tabelle bei einer vorherigen Beendigung des Systems **100** gespeichert wurde und der Ablauf **500** geht zu Schritt **510** zurück.

[0047] Wenn nicht, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, die Daten von der Caching-Einheit **170** in Schritt **550** zu lesen. Wenn bei einer erfindungsgemäßen Ausführungsform beispielsweise das OS Daten aus einem bestimmten Adressort in dem Speichergerät **160** lesen möchte, empfängt der Laufzeit-Controller eine Anfrage, Daten aus der Caching-Einheit **170** zu lesen. Wenn eine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** zu lesen, prüft der Laufzeit-Controller die Cache-Tabelle, ob die Daten an dem bestimmten Adressort in dem Speichergerät **160** in Schritt **552** in der Caching-Einheit **170** gecacht sind. Wenn keine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** zu lesen, geht der Ablauf zu Schritt **510** zurück.

[0048] In Schritt **554** prüft der Laufzeit-Controller, ob es einen Cache-Hit gibt, d. h. die Daten an dem bestimmten Adressort in dem Speichergerät **160** sind in der Caching-Einheit **170** gecacht. Wenn ja, liest der

Laufzeit-Controller die Daten aus der Caching-Einheit **170** und liefert die Daten in Schritt **556** an das OS zurück und der Ablauf **500** geht zu Schritt **510** zurück. Wenn nicht, sendet der Laufzeit-Controller in Schritt **558** einen Cache-Miss an das OS. Bei einer erfindungsgemäßen Ausführungsform greift der Laufzeit-Controller auf die Daten an dem bestimmten Adressort in dem Speichergerät **160** zu, wenn es einen Cache-Miss gibt, und liefert in Schritt **558** die Daten an das OS zurück und der Ablauf **500** geht zu Schritt **510** zurück.

[0049] Bei einer erfindungsgemäßen Ausführungsform schreibt oder aktualisiert der Laufzeit-Controller die gepackten Metadaten in der Caching-Einheit **170** nicht während der Laufzeit, wenn ein Write-through-Caching-Schema verwendet wird. Da die Daten in dem Speichergerät **160** und Caching-Einheit **170** immer synchronisiert sind, kann die Caching-Einheit **170** zurückgesetzt werden, wenn ein Fehlerereignis, wie z. B. ein Energieverlustereignis, auftritt. Das System **100** ist energieausfallsicher, da die Integrität der Daten in dem Speichergerät **160** sogar während eines Energieverlustereignisses aufrechterhalten wird.

[0050] Fig. 6A veranschaulicht ein Ablaufdiagramm **600** eines Write-back-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. In Schritt **610** prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, eine Cache-Line(s) der Caching-Einheit **170** zu aktualisieren. Wenn eine Anfrage zum Aktualisieren der Cache-Line(s) vorliegt, aktualisiert der Laufzeit-Controller die entsprechende(n) Cache-Line(s) mit den neuen Daten in Schritt **612**. In Schritt **614** aktualisiert der Laufzeit-Controller die gepackten Metadaten oder Zustandsinformationen, die mit der/den Cache-Line(s) verbunden sind, basierend auf den neuen zu schreibenden Daten in dem flüchtigen Speicher **142**. In Schritt **616** aktualisiert der Laufzeit-Controller die gepackten Metadaten oder Zustandsinformationen, die mit der/den Cache-Line(s) verbunden sind, basierend auf den neuen zu schreibenden Daten in der Caching-Einheit **170**. Bei einer weiteren Ausführungsform kopiert der Laufzeit-Controller in Schritt **616** die gepackten Metadaten oder Zustandsinformationen, die mit der/den Cache-Line(s) in dem flüchtigen Speicher **142** verbunden sind, in die entsprechenden gepackten Metadaten, die mit der/den Cache-Line(s) in dem logischen Segment gepackter Metadaten **401** der Caching-Einheit **170** verbunden sind. Der Ablauf **600** geht zu Schritt **610** zurück, nachdem Schritt **616** ausgeführt ist.

[0051] Wenn keine Anfrage zum Aktualisieren der Cache-Line(s) vorliegt, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, das System **100** sachgemäß in Schritt **620** herunterzufahren. Wenn eine Anfrage vorliegt, das System **100** sachgemäß herunterzufahren, kopiert der Laufzeit-Controller in einem optionalen Schritt **624** die Cache-Tabelle von dem flüchtigen Speicher **142** in die Caching-Einheit **170** und der Ablauf **600** geht zu Schritt **610** zurück. Wenn keine Anfrage vorliegt, das System **100** sachgemäß herunterzufahren, prüft der Laufzeit-Controller, ob das OS einen Flush-Befehl in Schritt **630** ausgegeben hat. Wenn das OS einen Flush-Befehl ausgegeben hat, flusht der Laufzeit-Controller in Schritt **632** jegliche flüchtigen Daten sowohl in dem Speichergerät **160** als auch in der Caching-Einheit **170**.

[0052] Wenn das OS keinen Flush-Befehl ausgegeben hat, prüft der Wiederherstellungs-Controller, ob irgendeine Benachrichtigung zum Hochfahren des Systems **100** in Schritt **640** vorliegt. Wenn ja, speichert der Wiederherstellungs-Controller die gepackten Metadaten in der Caching-Einheit **170** in Schritt **642** erneut in den flüchtigen Speicher **142** oder kopiert sie dorthin. Im optionalen Schritt **644** speichert der Wiederherstellungs-Controller die Cache-Tabelle in der Caching-Einheit **170** erneut in den flüchtigen Speicher **142** oder kopiert sie dorthin, wenn die Cache-Tabelle bei einer vorherigen Beendigung des Systems **100** gespeichert wurde, und der Ablauf **600** geht zu Schritt **610** zurück.

[0053] Wenn nicht, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, die Daten von der Caching-Einheit **170** in Schritt **650** zu lesen. Wenn eine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** zu lesen, prüft der Laufzeit-Controller die Cache-Tabelle, ob die Daten an dem bestimmten Adressort in dem Speichergerät **160** in Schritt **652** in der Caching-Einheit **170** gecacht sind. Wenn keine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** zu lesen, geht der Ablauf zu Schritt **610** zurück.

[0054] In Schritt **654** prüft der Laufzeit-Controller, ob es einen Cache-Hit gibt, d. h. die Daten an dem bestimmten Adressort in dem Speichergerät **160** sind in der Caching-Einheit **170** gecacht. Wenn ja, liest der Laufzeit-Controller die Daten aus der Caching-Einheit **170** und liefert die Daten in Schritt **656** an das OS zurück und der Ablauf **600** geht zu Schritt **610** zurück. Wenn nicht, sendet der Laufzeit-Controller in Schritt **658** einen Cache-Miss an das OS. Bei einer erfindungsgemäßen Ausführungsform greift der Laufzeit-Controller auf die Daten an dem bestimmten Adressort in dem Speichergerät **160** zu, wenn es einen Cache-Miss gibt, und liefert in Schritt **658** die Daten an das OS zurück, und der Ablauf **600** geht zu Schritt **610** zurück. Das Write-back-Caching-Schema von Fig. 6A erfordert einen zusätzlichen Schreibvorgang auf die Caching-Einheit **170**, um die gepackten Metadaten, die mit der/den Cache-Line(s) für jeden Cache-Line-Schreibvorgang für neue Daten verbunden sind, zu aktualisieren.

[0055] Fig. 6B veranschaulicht ein Ablaufdiagramm

660 eines Write-back-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. In Schritt **610** prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, eine Cache-Line(s) der Caching-Einheit **170** zu aktualisieren. Wenn eine Anfrage zum Aktualisieren der Cache-Line(s) vorliegt, aktualisiert der Laufzeit-Controller die entsprechende(n) Cache-Line(s) mit den neuen Daten in Schritt **612**. In Schritt **614** aktualisiert der Laufzeit-Controller die gepackten Metadaten oder Zustandsinformationen, die mit der/den Cache-Line(s) verbunden sind, basierend auf den neuen zu schreibenden Daten in dem flüchtigen Speicher **142**. In Schritt **615** zeichnet der Laufzeit-Controller die Cache-Line(s) basierend auf den neuen Daten in der Caching-Einheit **170** als eine ausstehende Aktualisierung der gepackten Metadaten, die mit den Cache-Lines verbunden sind, aus. Bei einer erfindungsgemäßen Ausführungsform zeichnet der Laufzeit-Controller die Cache-Line(s) aus, indem er die Cache-Line(s) zu einer Liste der ausstehenden Metadaten-Schreibvorgänge im flüchtigen Speicher **142** hinzufügt. Die Liste ausstehender Metadaten-Schreibvorgänge beinhaltet Cache-Line(s), die verbundene gepackte Metadaten aufweisen, die nicht zwischen dem flüchtigen Speicher **142** und der Caching-Einheit **170** synchronisiert wurden.

[0056] Wenn keine Anfrage zum Aktualisieren der Cache-Line(s) vorliegt, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, das System **100** sachgemäß in Schritt **620** herunterzufahren. Wenn eine Anfrage vorliegt, das System **100** sachgemäß herunterzufahren, schreibt der Laufzeit-Controller alle ausstehenden gepackten Metadaten in dem flüchtigen Speicher **142** in die gepackten Metadaten in der Caching-Einheit **170**. Bei einer erfindungsgemäßen Ausführungsform bestimmt der Laufzeit-Controller aus der Liste ausstehender Metadaten-Schreibvorgänge, welche Metadaten aktualisiert oder geschrieben werden sollen. Im optionalen Schritt **624** kopiert der Laufzeit-Controller die Cache-Tabelle von dem flüchtigen Speicher **142** in die Caching-Einheit **170** und der Ablauf **660** geht zu Schritt **610** zurück.

[0057] Wenn keine Anfrage vorliegt, das System **100** sachgemäß herunterzufahren, prüft der Laufzeit-Controller, ob das OS einen Flush-Befehl in Schritt **630** ausgegeben hat. Wenn ein Flush-Befehl ausgegeben wurde, aktualisiert der Laufzeit-Controller in Schritt **631** alle ausstehenden gepackten Metadaten in dem flüchtigen Speicher **142** in die gepackten Metadaten in der Caching-Einheit **170**. Bei einer weiteren erfindungsgemäßen Ausführungsform aktualisiert oder kopiert der Laufzeit-Controller in einer einzigen sequentiellen Schreiboperation in Schritt **631** die gesamten gepackten Metadaten von dem flüchtigen Speicher **142** in die Caching-Einheit **170**. In Schritt **632** flusht der Laufzeit-Controller jegliche flüchtigen Daten sowohl in dem Speichergerät **160** als auch in der Caching-Einheit **170**.

[0058] Wenn kein Flush-Befehl ausgegeben wurde, prüft der Wiederherstellungs-Controller, ob irgendeine Benachrichtigung zum Hochfahren des Systems **100** in Schritt **640** vorliegt. Wenn ja, speichert der Wiederherstellungs-Controller die gepackten Metadaten in der Caching-Einheit **170** in Schritt **642** erneut in den flüchtigen Speicher **142** oder kopiert sie dorthin. Im optionalen Schritt **644** speichert der Wiederherstellungs-Controller die Cache-Tabelle in der Caching-Einheit **170** erneut in den flüchtigen Speicher **142** oder kopiert sie dorthin, wenn die Cache-Tabelle bei einer vorherigen Beendigung des Systems **100** gespeichert wurde, und der Ablauf **660** geht zu Schritt **610** zurück.

[0059] Wenn nicht, prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** in Schritt **650** zu lesen. Wenn eine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** zu lesen, prüft der Laufzeit-Controller die Cache-Tabelle, ob die Daten an dem bestimmten Adressort in dem Speichergerät **160** in Schritt **652** in der Caching-Einheit **170** gecacht sind. Wenn keine Anfrage vorliegt, die Daten aus der Caching-Einheit **170** zu lesen, geht der Ablauf **660** zu Schritt **610** zurück.

[0060] In Schritt **654** prüft der Laufzeit-Controller, ob es einen Cache-Hit gibt, d. h. ob die Daten an dem bestimmten Adressort in dem Speichergerät **160** in der Caching-Einheit **170** gecacht sind. Wenn ja, liest der Laufzeit-Controller die Daten aus der Caching-Einheit **170** und liefert die Daten in Schritt **656** an das OS zurück, und der Ablauf **660** geht zu Schritt **610** zurück. Wenn nicht, sendet der Laufzeit-Controller in Schritt **658** einen Cache-Miss an das OS. Bei einer erfindungsgemäßen Ausführungsform greift der Laufzeit-Controller auf die Daten an dem bestimmten Adressort in dem Speichergerät **160** zu, wenn es einen Cache-Miss gibt, und liefert in Schritt **658** die Daten an das OS zurück, und der Ablauf **660** geht zu Schritt **610** zurück. Das Write-back-Caching-Schema von Fig. 6B erfordert einen optionalen zusätzlichen Schreibvorgang auf die Caching-Einheit **170**, um die gepackten Metadaten, die mit der/den Cache-Line(s) für jedes Ereignis zum Flushen oder Herunterfahren verbunden sind, zu aktualisieren.

[0061] Fig. 6C veranschaulicht ein Ablaufdiagramm **680** eines Write-back-Caching-Schemas in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform. Fig. 6C wird mit Bezug auf Fig. 6B erörtert, da der Ablauf **680** eine Variante des Ablaufs **660** ist. Alle Schritte in Ablauf **660**, mit Ausnahme von Schritt **631**, gelten für Ablauf **680**, und diese Schritte sollen hierin nicht wiederholt werden. In Ablauf **680** prüft der Laufzeit-Controller, nachdem ein Flush-Befehl vom OS in Schritt **630** empfangen wurde, ob angrenzende ausstehende Schreibvorgänge in der Lis-

te ausstehender Metadaten-Schreibvorgänge in Schritt **662** vorliegen. Zum Zwecke der Veranschaulichung wird angenommen, dass die Liste ausstehender Metadaten-Schreibvorgänge ausstehende Metadaten-Schreibvorgänge für sieben Cache-Lines (Cache-Lines 5, 6, 7, 9, 12, 13 und 45) aufweist.

[0062] Bei dem angenommenen Szenario geht der Ablauf **680** zu Schritt **664**, da Cache-Lines 5, 6 und 7 angrenzend sind und Cache-Lines 12 und 13 ebenfalls angrenzend sind. In Schritt **664** kombiniert der Laufzeit-Controller die Metadaten-Schreibvorgänge für Cache-Lines 5, 6 und 7 in einen einzigen Metadaten-Schreibvorgang. Er kombiniert ebenfalls die Metadaten-Schreibvorgänge für Cache-Lines 12 und 13 in einen weiteren einzigen Metadaten-Schreibvorgang. Daher hat der Laufzeit-Controller vier Metadaten-Schreibvorgänge (kombinierter Schreibvorgang von 5, 6 und 7, 9, kombinierter Schreibvorgang von 12 und 13 sowie 45) anstatt der ursprünglich sieben Metadaten-Schreibvorgänge. In Schritt **670** führt der Laufzeit-Controller die vier Metadaten-Schreibvorgänge von Schritt **664** aus.

[0063] Bei einem weiteren veranschaulichenden Beispiel wird angenommen, dass die Liste ausstehender Metadaten-Schreibvorgänge ausstehende Metadaten-Schreibvorgänge für fünf Cache-Lines (Cache-Lines 3, 9, 11, 14 und 45) aufweist. Bei dem angenommenen Szenario geht der Ablauf **680** zu Schritt **662**, um zu prüfen, ob es einen kleinen Abstand bei dem Adressort der ausstehenden zu schreibenden Cache-Lines gibt. Der Abstand zwischen den Cache-Lines wird als klein bezeichnet, wenn die Zeit, die benötigt wird, um die Cache-Lines gemeinsam zu schreiben, kürzer ist als die Zeit, die benötigt wird, bei einer erfindungsgemäßen Ausführungsform die Cache-Lines getrennt zu schreiben. Wenn beispielsweise die Zeit, die erforderlich ist, um die mit Cache-Lines 9, 10 und 11 verbundenen Metadaten zu aktualisieren, kürzer ist als die Zeit, die erforderlich ist, um die mit Cache-Lines 9 und 11 verbundenen Metadaten getrennt zu aktualisieren, wird der Abstand zwischen Cache-Lines 9 und 11 als klein angesehen. Obwohl die mit Cache-Line 10 verbundenen Metadaten nicht aktualisiert werden müssen, verringert eine Kombination der Metadaten-Aktualisierung der Cache-Lines die erforderliche Zeit, um die Cache-Lines bei einer erfindungsgemäßen Ausführungsform zu aktualisieren.

[0064] Bei dem angenommenen Szenario wird der Abstand zwischen Cache-Lines 9 und 11 und zwischen Cache-Lines 11 und 14 als klein angenommen und der Ablauf geht zu Schritt **668**. In Schritt **668** kombiniert der Laufzeit-Controller die Cache-Lines mit einem kleinen Abstand zwischen ihnen zu einem großen Metadaten-Cache-Schreibvorgang. Bei dem angenommenen Szenario beispielsweise kombiniert der Laufzeit-Controller eine Metadaten-Aktualisierung für Cache-Lines 9, 11 und 14 in eine einzige Metadaten-Aktualisierung für Cache-Lines 9–14, obwohl Cache-Lines 10, 12 und 13 nicht modifiziert werden müssen. In Schritt **670** führt der Laufzeit-Controller die kombinierten Metadaten-Schreibvorgänge von Schritt **664** aus und der Ablauf geht zu Schritt **634** in Ablauf **660**. Die Schritte **664** und **668** optimieren die Operationen, um die ausstehenden gepackten Metadaten in der Caching-Einheit **170** zu aktualisieren. In Ablauf **680** bei einer weiteren erfindungsgemäßen Ausführungsform kann nur einer der Schritte **662** und **664** sowie der Schritte **666** und **668** ausgeführt werden. Für einen Fachmann ist es sofort selbstverständlich, dass weitere Optimierungen durchgeführt werden können, um die Zeit zum Aktualisieren der ausstehenden Metadaten-Aktualisierungen in der Caching-Einheit **170** zu verringern, ohne die erfindungsgemäßen Arbeiten zu beeinträchtigen.

[0065] Die in Fig. 6A, Fig. 6B und Fig. 6C veranschaulichten Write-back-Caching-Schemata sollen als nicht einschränkend verstanden werden. Für einen Fachmann ist es sofort selbstverständlich, dass verschiedene Kombinationen oder Modifikationen der Schritte durchgeführt werden können, ohne die erfindungsgemäßen Arbeiten zu beeinträchtigen. Ein Benutzer des Systems **100** kann sich entscheiden, eines der drei Write-back-Caching-Schemata von Fig. 6A, Fig. 6B und Fig. 6C zu verwenden, und kann ebenfalls jede Kombination der drei Write-back-Schemata von Fig. 6A, Fig. 6B und Fig. 6C verwenden.

[0066] Fig. 7 veranschaulicht ein Ablaufdiagramm **700** eines Verfahrens, um Daten in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform in eine Cache-Line der Caching-Einheit **170** einzufügen. In Schritt **710** prüft der Laufzeit-Controller, ob eine Anfrage vorliegt, Daten in die Cache-Line(s) der Caching-Einheit **170** einzufügen. Wenn bei einer erfindungsgemäßen Ausfühnmgsform beispielsweise das OS Daten auf einen bestimmten Adressort in dem Speichergerät **160** schreiben möchte, prüft der Laufzeit-Controller die Cache-Tabelle, ob die Daten an dem bestimmten Adressort in dem Speichergerät **160** in die Caching-Einheit **170** gecacht sind. Wenn es keinen Cache-Hit gibt, d. h. die Daten an dem bestimmten Adressort nicht in der Caching-Einheit **170** gecacht sind, kann der Laufzeit-Controller eine Anfrage empfangen, um die Daten in die Cache-Line(s) der Caching-Einheit **170** einzufügen.

[0067] Wenn keine Anfrage vorliegt, Daten in die Cache-Line(s) einzufügen, endet der Ablauf. Wenn eine Anfrage vorliegt, Daten in die Cache-Line(s) einzufügen, prüft der Laufzeit-Controller, ob es irgendwelche freien Cache-Lines in der Caching-Einheit **170** in Schritt **720** gibt. Bei einer erfindungsgemäßen Ausführungsform werden alle unbenutzten Cache-Lines in der Caching-Einheit **170** als freie Cache-Lines

ausgezeichnet oder markiert. Bei einer weiteren erfindungsgemäßen Ausführungsform wird ein fester Anteil der unbenutzten Cache-Lines in der Caching-Einheit **170** als freie Cache-Lines ausgezeichnet oder markiert. Bei einer Ausführungsform beispielsweise kann der Laufzeit-Controller fünf Cache-Lines der Caching-Einheit **170** als freie Cache-Lines auszeichnen. Wenn es keine freien Cache-Lines gibt, wählt der Laufzeit-Controller basierend auf einer Richtlinie zum Entfernen in Schritt **722** eine oder mehr Cache-Lines der Caching-Einheit **170** zum Entfernen aus. Die Richtlinie zum Entfernen beinhaltet, ist aber nicht darauf beschränkt, dass die zuletzt benutzte(n) Cache-Line(s) entfernt wird/werden, dass die erste Cache-Line der Caching-Einheit **170** entfernt wird oder dergleichen.

[0068] In Schritt **724** wird/werden die ausgewählte(n) Cache-Line(s) durch den Laufzeit-Controller entfernt. Bei einer erfindungsgemäßen Ausführungsform entfernt der Laufzeit-Controller die ausgewählte(n) Cache-Line(s), indem die gecachten Daten in der/den ausgewählten Cache-Line(s) in das Speichergerät **160** geschrieben werden, sofern sie noch nicht synchronisiert wurden. In Schritt **726** markiert der Laufzeit-Controller die entfernte(n) Cache-Line(s) als freie Cache-Line(s) oder zeichnet sie so aus, und der Ablauf geht zu Schritt **730**. Wenn es freie Cache-Line(s) gibt, wählt der Laufzeit-Controller eine oder mehrere freie Cache-Line(s) der Caching-Einheit **170** aus, um die Daten, die in Schritt **730** geschrieben werden sollen, zu cachen. Die Auswahl-Richtlinie der freien Cache-Line(s) beinhaltet, ist aber nicht beschränkt auf, erste verfügbare freie Cache-Line, zuletzt verwendete freie Cache-Line und dergleichen. In Schritt **740** schreibt der Laufzeit-Controller die Daten auf die ausgewählte(n) freie(n) Cache-Line(s).

[0069] In Schritt **750** aktualisiert der Laufzeit-Controller die gepackten Metadaten oder Zustandsinformationen, die mit der/den ausgewählten Cache-Line(s) verbunden sind, basierend auf den neuen Daten in dem flüchtigen Speicher **142**. Nach Schritt **750** kann der Ablauf **700** einen optionalen Schritt **760** ausführen, bei dem der Laufzeit-Controller die Cache-Line(s) als eine ausstehende Aktualisierung der gepackten Metadaten, die mit den Cache-Lines verbunden sind, basierend auf den neuen Daten in der Caching-Einheit **170** auszeichnet, wenn das Write-back-Caching-Schema von Ablauf **660** oder **680** verwendet wird, oder er kann einen optionalen Schritt **770** ausführen, bei dem der Laufzeit-Controller die gepackten Metadaten oder Zustandsinformationen, die mit der/den ausgewählten Cache-Line(s) verbunden sind, basierend auf den neuen Daten in der Caching-Einheit **170** aktualisiert, wenn das Write-back-Caching-Schema von Ablauf **600** verwendet wird. Der Ablauf endet, nachdem entweder optionaler Schritt **760** oder **770** ausgeführt ist.

[0070] Wenn Cache-Line(s) entfernt werden, wird ein sofortiges Aktualisieren der gepackten Metadaten, die mit der/den Cache-Line(s) in der Caching-Einheit **170** verbunden sind, erforderlich, da ein Fehlerereignis, wie z. B. ein Energieausfall des Systems **100**, ein Datenintegritätsproblem in dem Speichergerät **160** und der Caching-Einheit **170** hervorrufen kann. Somit ist beim Entfernen von Cache-Line(s) ein Metadaten-Schreibvorgang, die mit der/den Cache-Line(s) in der Caching-Einheit **170** verbunden sind, nach jedem Entfernen erforderlich. Jedoch führt ein Durchführen eines zusätzlichen Metadaten-Schreibvorganges nach jedem Entfernen der Cache-Line(s) zu Mehraufwand. Um den Mehraufwand zu vermeiden, beinhaltet das in Fig. 7 erörterte Verfahren zum Einfügen von Daten in eine Cache-Line(s) der Caching-Einheit **170**, dass neue Daten in freie Cache-Line(s) anstatt in eine Cache-Line(s) mit gecachten Daten eingefügt werden.

[0071] Zum Zwecke der Veranschaulichung wird beispielsweise angenommen, dass der Laufzeit-Controller eine Anfrage zum Einfügen von Daten für LBA 1 des Speichergeräts **160** empfängt. Es wird angenommen, dass Cache-Line 4 die Daten von LBA 5 des Speichergeräts **160** speichert. Wenn ein Fehlerereignis auftritt, nachdem die Daten von LBA 1 des Speichergeräts **160** auf Cache-Line 4 geschrieben wurden, aber bevor die mit Cache-Line 4 verbundenen Metadaten aktualisiert wurden, wird das System **100** bei einem Neustart- oder Rebootereignis erkennen, dass Cache-Line 4 über die Daten von LBA 5 basierend auf den mit Cache-Line 4 verbundenen Metadaten verfügt. Dies ist jedoch falsch, da die Cache-Line 4 mit den Daten von LBA 1 des Speichergeräts **160** aktualisiert wurde.

[0072] Dadurch, dass neue Daten in freie Cache-Lines, wie in dem Ablauf **700** von Fig. 7 beschrieben, eingefügt werden, beeinträchtigt ein auftretendes Fehlerereignis die Datenintegrität des Speichergeräts **160** und Caching-Einheit **170** nicht. Wenn zum Zwecke der Veranschaulichung beispielsweise der Laufzeit-Controller eine Anfrage empfängt, um die Daten von LBA 1 des Speichergeräts **160** in die Caching-Einheit **170** einzufügen, wählt die Laufzeit eine freie Cache-Line aus, um die Daten von LBA 1 des Speichergeräts **160** zu cachen. Wenn ein Fehlerereignis auftritt, nachdem die freie Cache-Line mit den Daten von LBA 1 des Speichergeräts **160** aktualisiert wurde, aber bevor die mit der freien Cache-Line verbundenen Metadaten aktualisiert werden, beeinträchtigt das Fehlerereignis die Datenintegrität in dem Speichergerät **160** und der Caching-Einheit **170** nicht. Da es sich um ein Fehlerereignis handelt und kein Flush-Ereignis aufgetreten ist, können die neuen Daten verworfen werden, ohne das System **100** zu beeinträchtigen.

[0073] Das Write-through-Caching-Schema ist nicht

auf den in Fig. 5 gezeigten Algorithmus beschränkt. Bei einer weiteren erfindungsgemäßen Ausführungsform kann das Write-through-Caching-Schema einen der in Fig. 6A, Fig. 6B und Fig. 6C gezeigten Write-back-Caching-Algorithmen verwenden. Das Write-back-Caching-Schema kann einen der in Fig. 6A, Fig. 6B und Fig. 6C gezeigten Write-back-Caching-Algorithmen und die in Fig. 7 gezeigten Algorithmen verwenden. Wenn das Write-through-Caching-Schema einen der in Fig. 6A, Fig. 6B und Fig. 6C gezeigten Write-back-Caching-Algorithmen und die in Fig. 7 gezeigten Algorithmen verwendet, kann der Write-through-Cache auch über unsachgemäße Beendigungen warm gehalten werden.

[0074] Fig. 8A und Fig. 8B veranschaulichen einen Pseudo-Code **800** und **850**, um ein Write-back-Caching-Schema in Übereinstimmung mit einer erfindungsgemäßen Ausführungsform zu implementieren. Zum Zwecke der Veranschaulichung wird ein HDD verwendet, um das Speichergerät **160** beispielhaft darzustellen, und ein SSD wird verwendet, um die Caching-Einheit **170** beispielhaft darzustellen. Für einen Fachmann werden die Arbeiten des Pseudo-Codes **800** und **850** sofort selbstverständlich sein, und die Arbeiten des Pseudo-Codes **800** und **850** sollen nicht ausführlich erörtert werden.

[0075] Obwohl Beispiele der Ausführungsformen des offenbarten Gegenstands beschrieben sind, ist es für einen Fachmann sofort selbstverständlich, dass viele andere Verfahren zum Implementieren des offenbarten Gegenstands alternativ verwendet werden können. Bei der vorstehenden Beschreibung sind verschiedene Ausführungsformen des offenbarten Gegenstands beschrieben worden. Zum Zwecke der Erklärung wurden spezifische Anzahlen, Systeme und Konfigurationen dargelegt, um ein gründliches Verständnis des Gegenstands bereitzustellen. Es ist jedoch einem Fachmann, der den Vorteil dieser Offenbarung hat, selbstverständlich, dass der Gegenstand ohne die spezifischen Details umgesetzt werden kann. In anderen Fällen wurden wohl bekannte Merkmale, Komponenten oder Module weggelassen, vereinfacht, kombiniert oder aufgeteilt, um den offenbarten Gegenstand nicht zu verschleiern.

[0076] Der hierin verwendete Begriff „arbeitet" bedeutet, dass das Gerät, System, Protokoll etc. in der Lage ist oder dazu ausgelegt ist, für seine gewünschte Funktionalität zu arbeiten, wenn sich das Gerät oder System in einem ausgeschalteten Zustand befindet. Verschiedene Ausführungsformen des offenbarten Gegenstands können in Hardware, Firmware, Software oder Kombinationen davon implementiert sein, und können unter Bezugnahme auf oder in Verbindung mit Programmcode, wie z. B. Befehlen, Funktionen, Verfahrensweisen, Datenstrukturen, Logik, Anwendungsprogrammen, Designdarstellungen oder Formaten zur Simulation, Emulation und Herstellung eines Designs beschrieben werden, die, wenn von einer Maschine darauf zugegriffen wird, dazu führen, dass die Maschine Aufgaben ausführt, wobei abstrakte Datentypen oder hardwarenaher Zusammenhang definiert wird oder ein Ergebnis erzeugt wird.

[0077] Die in den Figuren gezeigten Techniken können unter Verwendung von Code und Daten implementiert werden, die auf einem oder mehr EDV-Geräten, wie z. B. Universalrechnern oder EDV-Geräten, gespeichert und ausgeführt werden. Solche EDV-Geräte speichern und kommunizieren (intern und mit anderen EDV-Geräten über ein Netzwerk) Code und Daten unter Verwendung maschinenlesbarer Medien, wie z. B. maschinenlesbare Speichermedien (z. B. Magnetplatten, optische Platten, Direktzugriffsspeicher, Festspeicher, Flash-Speicher-Geräte, Phasenwechselspeicher) und maschinenlesbarer Kommunikationsmedien (z. B. elektrische, optische, akustische oder andere Formen sich ausbreitender Signale – wie z. B. Trägersignale, Infrarotsignale, digitale Signale etc.).

[0078] Während der offenbarte Gegenstand unter Bezugnahme auf veranschaulichende Ausführungsformen beschrieben wurde, soll diese Beschreibung nicht einschränkend ausgelegt werden. Verschiedene Modifikationen der veranschaulichenden Ausführungsformen sowie weiterer Ausführungsformen des Gegenstands, die Fachleuten, die der offenbarte Gegenstand betrifft, offensichtlich sind, sollen als im Umfang des offenbarten Gegenstands liegend erachtet werden.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Diese Liste der vom Anmelder aufgeführten Dokumente wurde automatisiert erzeugt und ist ausschließlich zur besseren Information des Lesers aufgenommen. Die Liste ist nicht Bestandteil der deutschen Patent- bzw. Gebrauchsmusteranmeldung. Das DPMA übernimmt keinerlei Haftung für etwaige Fehler oder Auslassungen.

Zitierte Nicht-Patentliteratur

- Institute of Electrical and Electronics Engineers (IEEE) 802.11 **[0024]**

Patentansprüche

1. Verfahren, umfassend:
Cachen von Daten eines ersten Geräts in eine oder mehrere einer Vielzahl von Cache-Lines eines zweiten Geräts, wobei keine mit irgendeiner der Cache-Lines verbundenen Zustandsinformation während des Cachens der Daten atomar mit den Daten in dem zweiten Gerät gespeichert werden soll.

2. Verfahren nach Anspruch 1, wobei ein Cachen der Daten des ersten Geräts in die eine oder mehrere Cache-Lines des zweiten Geräts umfasst:
Bestimmen von Zustandsinformationen, die mit der Vielzahl von Cache-Lines verbunden sind, wenn irgendeine der Vielzahl von Cache-Lines als unbenutzt ausgezeichnet ist;
wenn nicht,
Entfernen zumindest einer der Vielzahl von Cache-Lines und
Auszeichnen der zumindest einen entfernten Cache-Line als unbenutzt in den Zustandsinformationen;
Auswählen einer oder mehr Cache-Lines, die als unbenutzt in den Zustandsinformationen ausgezeichnet ist, und
Cachen der Daten des ersten Geräts in die ausgewählte eine oder mehrere Cache-Lines.

3. Verfahren nach Anspruch 1, weiter umfassend ein Speichern von Zustandsinformationen, die mit einer oder mehreren Cache-Lines verbunden sind, in einen Speicher, wenn die Daten des ersten Geräts in die eine oder mehrere Cache-Lines des zweiten Geräts gecacht sind.

4. Verfahren nach Anspruch 3, weiter umfassend:
Aktualisieren der gecachten Daten in der einen oder mehreren Cache-Lines mit neuen Daten, wenn die neuen Daten auf die eine oder mehrere Cache-Lines geschrieben werden sollen, und
Aktualisieren der Zustandsinformationen, die mit der einen oder mehreren Cache-Lines verbunden sind, basierend auf den neuen Daten in dem Speicher.

5. Verfahren nach Anspruch 4, weiter umfassend:
Kopieren aller Zustandsinformationen im Speicher auf das zweite Gerät, wenn einer der folgenden Befehle empfangen wird: ein Neustartbefehl, ein Befehl zur Beendigung, ein Hibernate-Befehl oder Standby-Befehl.

6. Verfahren nach Anspruch 4, weiter umfassend ein Zurücksetzen des zweiten Geräts, wenn ein Stromausfall oder Systemfehler aufgetreten ist.

7. Verfahren nach Anspruch 4, wobei das zweite Gerät eine Kopie der Zustandsinformationen umfasst, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, weiter umfassend, dass die Zustandsinformationen, die mit der einen oder mehreren Cache-Lines verbunden sind, basierend auf den neuen Daten in der Kopie der Zustandsinformationen in dem zweiten Gerät in Antwort auf ein Aktualisieren der gecachten Daten aktualisiert werden.

8. Verfahren nach Anspruch 4, wobei das zweite Gerät eine Kopie der Zustandsinformationen umfasst, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, weiter umfassend, dass die eine oder mehrere Cache-Lines als eine ausstehende Aktualisierung der Zustandsinformationen, die mit der einen oder mehreren Cache-Lines verbunden sind, basierend auf den neuen Daten in der Kopie der Zustandsinformationen in dem zweiten Gerät in Antwort auf ein Aktualisieren der gecachten Daten ausgezeichnet werden.

9. Verfahren nach Anspruch 8, weiter umfassend, dass die Zustandsinformationen, die mit der einen oder mehreren ausgezeichneten Cache-Lines in der Kopie der Zustandsinformationen in dem zweiten Gerät verbunden sind, aktualisiert werden, wenn einer der folgenden Befehle empfangen wird, ein Flush-Befehl, Neustartbefehl, Befehl zur Beendigung, Hibernate-Befehl oder Standby-Befehl.

10. Verfahren nach Anspruch 4, weiter umfassend, dass die Zustandsinformationen, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, auf das zweite Gerät kopiert werden, wenn ein Flush-Befehl empfangen wird.

11. Verfahren nach Anspruch 10, wobei ein Kopieren der Zustandsinformationen, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, auf das zweite Gerät umfasst, dass die Anzahl von Kopieroperationen optimiert wird.

12. Vorrichtung, umfassend:
einen NAND-Flash-Speicher mit einem ersten logischen Segment mit einer Vielzahl von Cache-Lines und einem zweiten logischen Segment mit einer Vielzahl von Metadaten, wobei jede der Metadaten mit einer entsprechenden der Vielzahl von Cache-Lines verbunden ist, wobei das zweite logische Segment keinerlei Metadaten aufweist, und
einen Controller, um Daten eines Speichergeräts in eine oder mehrere Cache-Lines zu cachen.

13. Vorrichtung nach Anspruch 12, wobei jede der Metadaten des zweiten logischen Segments zumindest eine von einer Sequenznummer, eine von einer logischen Blockadresse, eine von einer Zustandsinformation und Verankerungsinformationen gecachter Daten in ihrer entsprechenden verbundenen Cache-Line umfasst.

14. Vorrichtung nach Anspruch 12, wobei die

Vielzahl von Cache-Lines eine Vielzahl von freien Cache-Lines umfasst und wobei der Controller weiter:
bestimmen soll, ob alle freien Cache-Lines benutzt werden; wenn ja,
eine oder mehrere ausgewählte Cache-Lines entfernen soll, wobei die Auswahl auf einer Richtlinie zum Entfernen basiert;
die entfernte eine oder mehrere ausgewählte Cache-Lines zu den freien Cache-Lines hinzufügen soll, und wobei der Controller zum Cachen der Daten des Speichergeräts in die eine oder mehrere Cache-Lines die Daten des Speichergeräts in die eine oder mehrere freie Cache-Lines cachen soll.

15. Vorrichtung nach Anspruch 12, weiter umfassend einen Speicher, um die Vielzahl von Metadaten zu speichern, und wobei der Controller weiter:
die gecachten Daten der einen oder mehrerer Cache-Lines mit neuen Daten aktualisieren soll, wenn die neuen Daten auf die eine oder mehrere Cache-Lines geschrieben werden sollen, und
die Metadaten, die mit der einen oder mehreren Cache-Lines verbunden sind, basierend auf neuen Daten in dem Speicher aktualisieren soll.

16. Vorrichtung nach Anspruch 15, wobei der Controller weiter:
das Speichergerät mit den neuen Daten aktualisieren soll und
die Metadaten, die mit der einen oder mehreren Cache-Lines in dem NAND-Flash-Speicher verbunden sind, mit den Metadaten, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, überschreiben soll, wenn die Vorrichtung rebootet, beendet oder in einen Schlafzustand versetzt werden soll.

17. Vorrichtung nach Anspruch 15, wobei der Speicher weiter eine Cache-Tabelle speichern soll, die Linkinformationen zwischen den Daten in dem Speichergerät und der Vielzahl von Cache-Lines umfasst, und wobei der Controller weiter die Cache-Tabelle in dem Speicher in den NAND-Flash-Speicher kopieren soll, wenn die Vorrichtung rebootet, geflusht, beendet oder in einen Schlafzustand versetzt werden soll.

18. Vorrichtung nach Anspruch 15, wobei der Controller die Metadaten, die mit der einen oder mehreren Cache-Lines verbunden sind, basierend auf den neuen Daten in dem NAND-Flash-Speicher aktualisieren soll, wenn die neuen Daten auf die eine oder mehrere Cache-Lines als Antwort auf ein Aktualisieren der gecachten Daten geschrieben werden sollen.

19. Vorrichtung nach Anspruch 15, wobei der Controller weiter die eine oder mehrere Cache-Lines zu einer Liste hinzufügen soll, und wobei die Liste anzeigen soll, dass die Metadaten, die mit der einen oder mehreren Cache-Lines in dem NAND-Flash-Speicher verbunden sind, ausstehen, um mit den Metadaten, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, aktualisiert zu werden.

20. Vorrichtung nach Anspruch 19, wobei der Controller weiter die Metadaten, die mit der einen oder mehreren Cache-Lines in dem NAND-Flash-Speicher verbunden sind, mit den Metadaten, die mit der einen oder mehreren Cache-Lines in dem Speicher verbunden sind, aktualisieren soll, wenn die Vorrichtung ausgeschaltet oder rebootet werden soll oder wenn der NAND-Flash-Speicher geflusht werden soll.

21. Vorrichtung nach Anspruch 14, wobei der Controller weiter die Metadaten in dem nicht flüchtigen Medium mit den Metadaten in dem Speicher überschreiben soll, wenn ein von einem auf dem System ablaufenden Betriebssystem (operating system, OS) ausgegebener Flush-Befehl empfangen wird.

22. Vorrichtung nach Anspruch 20, wobei der Controller weiter die Metadaten in dem Speicher mit den Metadaten in dem NAND-Flash-Speicher erneut speichern soll, wenn die Vorrichtung hochgefahren oder neu gestartet werden soll.

23. Vorrichtung nach Anspruch 17, wobei der Controller weiter die Cache-Tabelle von dem NAND-Flash-Speicher in den Speicher kopieren soll, wenn die Vorrichtung hochgefahren oder neu gestartet werden soll.

24. Vorrichtung nach Anspruch 12, wobei das Speichergerät entweder eine Festplatte, ein Bandlaufwerk, eine Compact-Disk, eine Zip-Disk, ein Flash-Speicher oder ein Festkörperlaufwerk ist.

25. Vorrichtung nach Anspruch 12, wobei der Controller ein Blockspeichertreiber ist.

26. Computerlesbares Speichermedium mit darauf gespeicherten Befehlen, die, wenn sie ausgeführt werden, dafür sorgen, dass ein Prozessor das folgende Verfahren durchführt:
Write-back- oder Write-through-Caching von Daten eines Speichergeräts in eine oder mehrere einer Vielzahl von Cache-Lines einer Caching-Einheit, wobei keine Metadaten, die mit irgendeiner der Cache-Lines verbunden sind, atomar in die Caching-Einheit geschrieben werden sollen, wenn die Daten gecacht werden.

27. Medium nach Anspruch 26, wobei ein Write-back- oder Write-through-Caching der Daten des Speichergeräts in die eine oder mehrere Cache-Lines der Caching-Einheit umfasst:
Bestimmen, ob irgendeine der Vielzahl von Cache-Li-

nes als unbenutzt ausgezeichnet ist;
wenn nicht,
Entfernen zumindest einer der Vielzahl von Cache-Lines und
Auszeichnen der zumindest einen entfernten Cache-Line als unbenutzt;
Auswählen einer oder mehrerer Cache-Lines, die als unbenutzt ausgezeichnet sind, und
Cachen der Daten des Speichergeräts in die ausgewählte eine oder mehrere Cache-Lines.

28. Medium nach Anspruch 26, wobei ein Write-through-Caching der Daten umfasst:
Bestimmen, dass die eine oder mehrere Cache-Lines der Caching-Einheit mit neuen Daten aktualisiert werden soll;
Aktualisieren der einen oder mehreren bestimmten Cache-Lines mit den neuen Daten;
Aktualisieren von Metadaten, die mit der einen oder mehreren bestimmten Cache-Lines verbunden sind, basierend zumindest auf den neuen Daten in einem Speicher, und
Aktualisieren des Speichergeräts mit den neuen Daten.

29. Medium nach Anspruch 26, wobei ein Write-back-Caching der Daten umfasst:
Bestimmen, dass die eine oder mehrere Cache-Lines der Caching-Einheit mit neuen Daten aktualisiert werden soll;
Aktualisieren von Metadaten, die mit der einen oder mehreren bestimmten Cache-Lines verbunden sind, basierend zumindest auf den neuen Daten in einem Speicher;
Aktualisieren der einen oder von mehreren bestimmten Cache-Lines mit den neuen Daten und
Markieren der einen oder von mehreren bestimmten Cache-Lines, wobei das Markieren anzeigen soll, dass Metadaten, die mit der einen oder mehreren bestimmten Cache-Lines in der Caching-Einheit verbunden sind, mit den Metadaten, die mit der einen oder mehreren bestimmten Cache-Lines in dem Speicher verbunden sind, aktualisiert werden sollen.

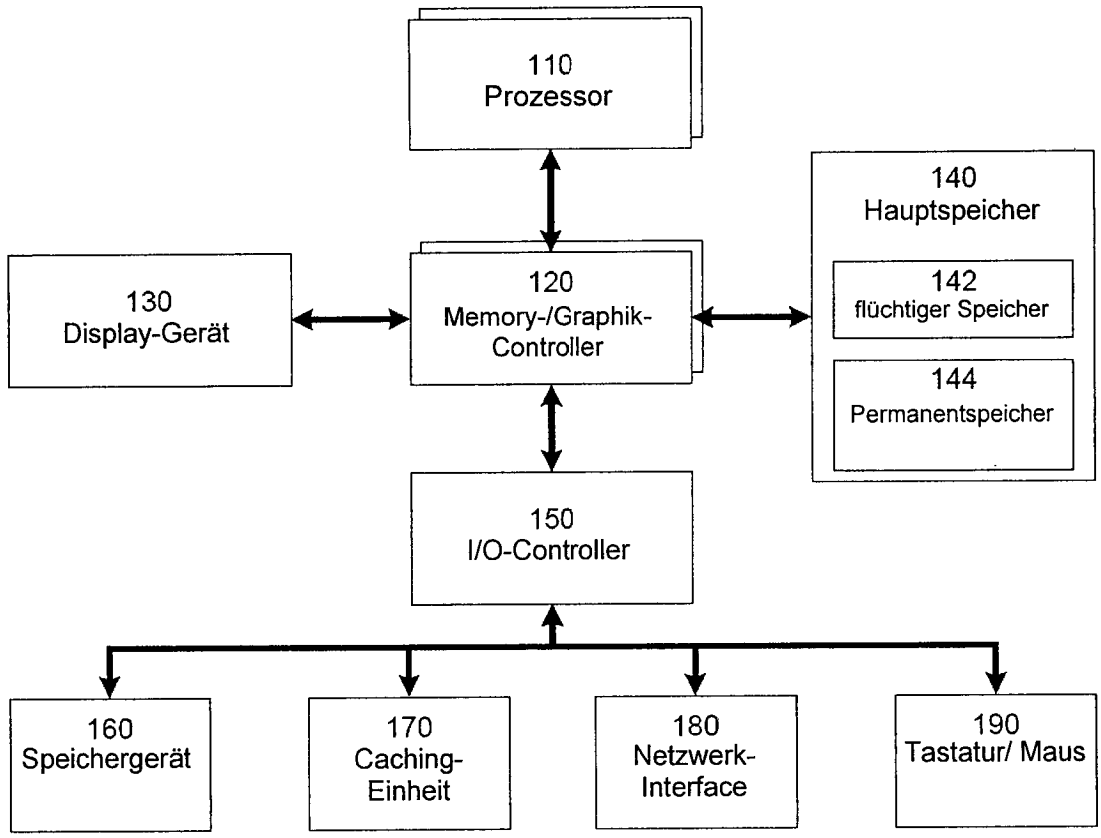
30. Medium nach Anspruch 26, wobei ein Write-back-Caching der Daten werter umfasst:
Aktualisieren der Metadaten, die mit der einen oder mehreren bestimmten Cache-Lines in der Caching-Einheit verbunden sind, mit den Metadaten, die mit der einen oder mehreren bestimmten Cache-Lines in dem Speicher verbunden sind, wenn ein Befehl zum Herunterfahren, Rebnooten oder Flushen empfangen wird.

31. System, umfassend:
einen NAND-Flash-Speicher mit einem ersten logischen Segment mit einer Vielzahl von Cache-Lines und einem zweiten logischen Segment mit einer Vielzahl von Metadaten, wobei jede der Metadaten mit einer entsprechenden der Vielzahl von Cache-Lines verbunden ist, wobei das zweite logische Segment keinerlei Metadaten aufweist, und
einen Memory-Controller zum Write-back oder Write-through von Cache-Daten eines Speichermediums in eine oder mehrere Cache-Lines des ersten logischen Segments des NAND-Flash-Speichers, wobei keine Metadaten atomar in das erste logische Segment geschrieben werden sollen, wenn die Daten gecacht werden.

32. System nach Anspruch 31, wobei jede der Metadaten des zweiten logischen Segments zumindest eine von einer Sequenznummer, eine von einer logischen Blockadresse, eine von einer Zustandsinformation und Verankerungsinformationen gecachter Daten in ihrer entsprechenden verbundenen Cache-Line umfasst.

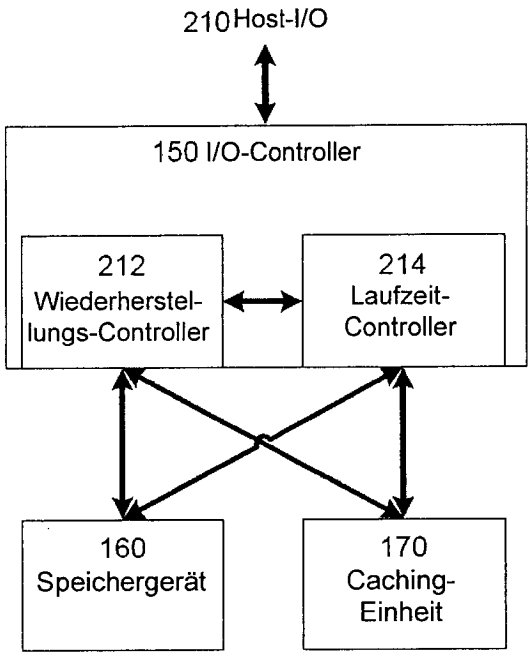
Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen



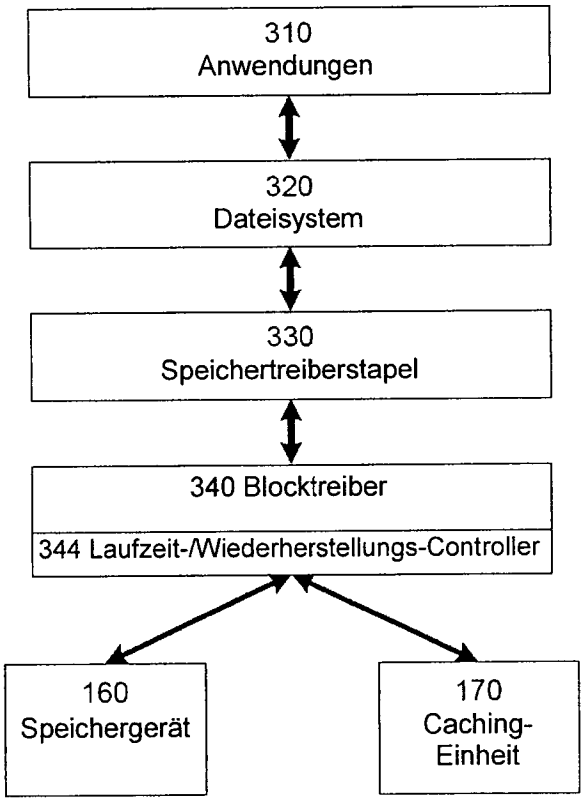
100

FIG. 1



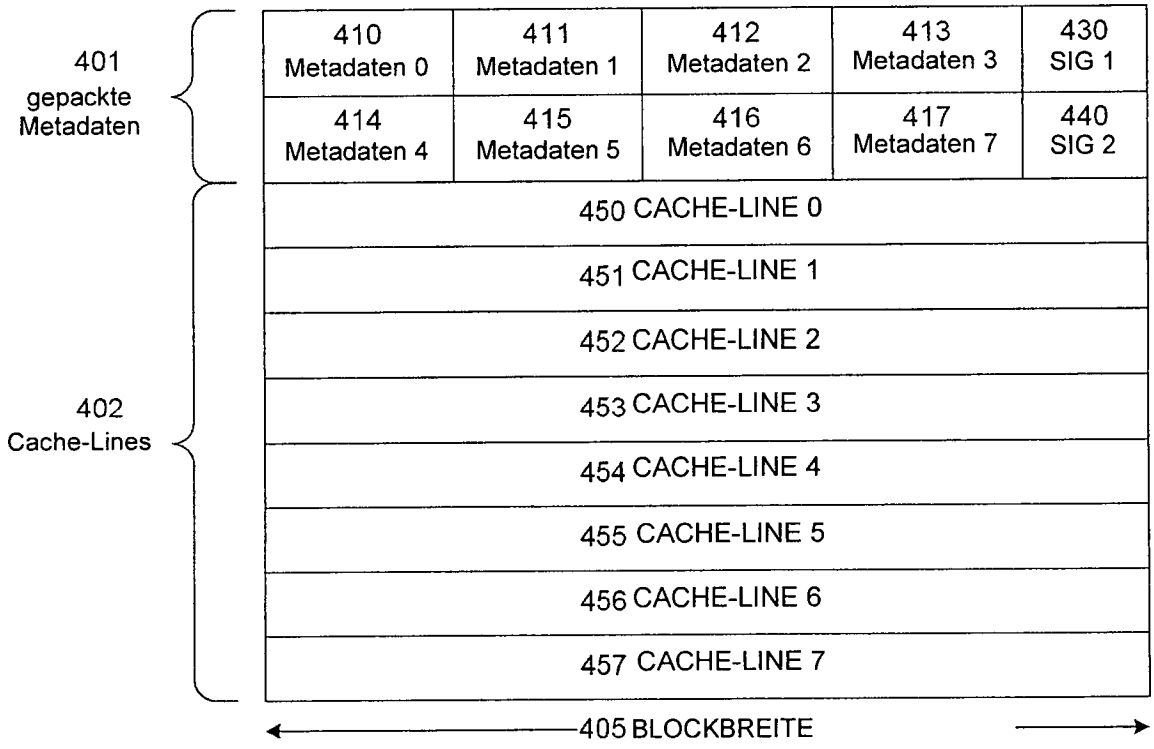
200

FIG. 2



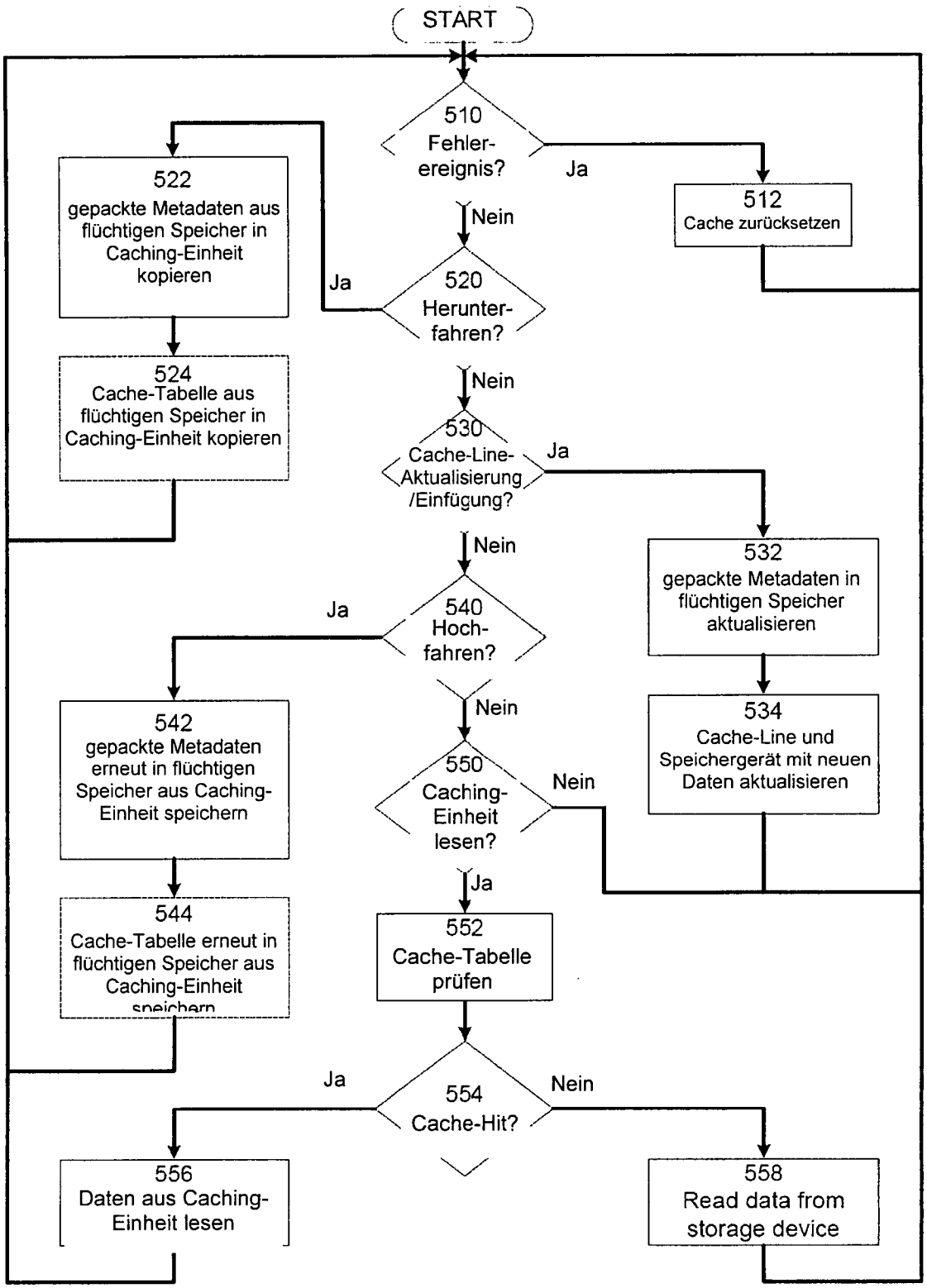
300

FIG. 3



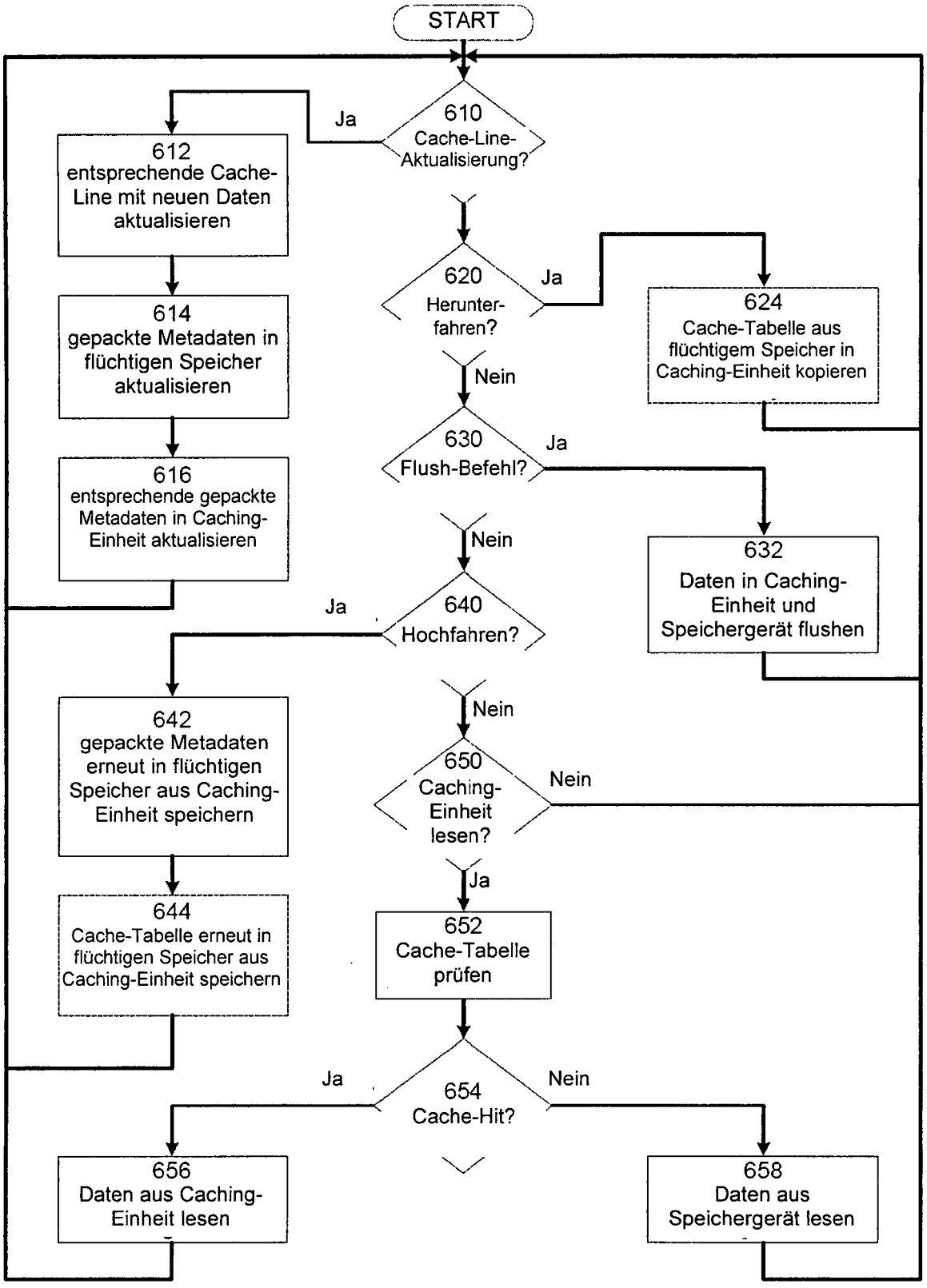
400

FIG. 4


START
510
Fehler-ereignis?
Ja
512
Cache zurücksetzen
Nein
520
Herunter-fahren?
Ja
522
gepackte Metadaten aus flüchtigen Speicher in Caching-Einheit kopieren
524
Cache-Tabelle aus flüchtigen Speicher in Caching-Einheit kopieren
Nein
530
Cache-Line-Aktualisierung /Einfügung?
Ja
532
gepackte Metadaten in flüchtigen Speicher aktualisieren
534
Cache-Line und Speichergerät mit neuen Daten aktualisieren
Nein
540
Hoch-fahren?
Ja
542
gepackte Metadaten erneut in flüchtigen Speicher aus Caching-Einheit speichern
544
Cache-Tabelle erneut in flüchtigen Speicher aus Caching-Einheit speichern
Nein
550
Caching-Einheit lesen?
Nein
Ja
552
Cache-Tabelle prüfen
Ja
554
Cache-Hit?
Nein
556
Daten aus Caching-Einheit lesen
558
Read data from storage device

500

FIG. 5


START
610
Cache-Line-Aktualisierung?
Ja
612
entsprechende Cache-Line mit neuen Daten aktualisieren
614
gepackte Metadaten in flüchtigen Speicher aktualisieren
616
entsprechende gepackte Metadaten in Caching-Einheit aktualisieren
620
Herunter-fahren?
Ja
624
Cache-Tabelle aus flüchtigem Speicher in Caching-Einheit kopieren
Nein
630
Flush-Befehl?
Ja
632
Daten in Caching-Einheit und Speichergerät flushen
Nein
640
Hochfahren?
Ja
642
gepackte Metadaten erneut in flüchtigen Speicher aus Caching-Einheit speichern
644
Cache-Tabelle erneut in flüchtigen Speicher aus Caching-Einheit speichern
Nein
650
Caching-Einheit lesen?
Nein
Ja
652
Cache-Tabelle prüfen
654
Cache-Hit?
Ja
Nein
656
Daten aus Caching-Einheit lesen
658
Daten aus Speichergerät lesen

600

FIG. 6A

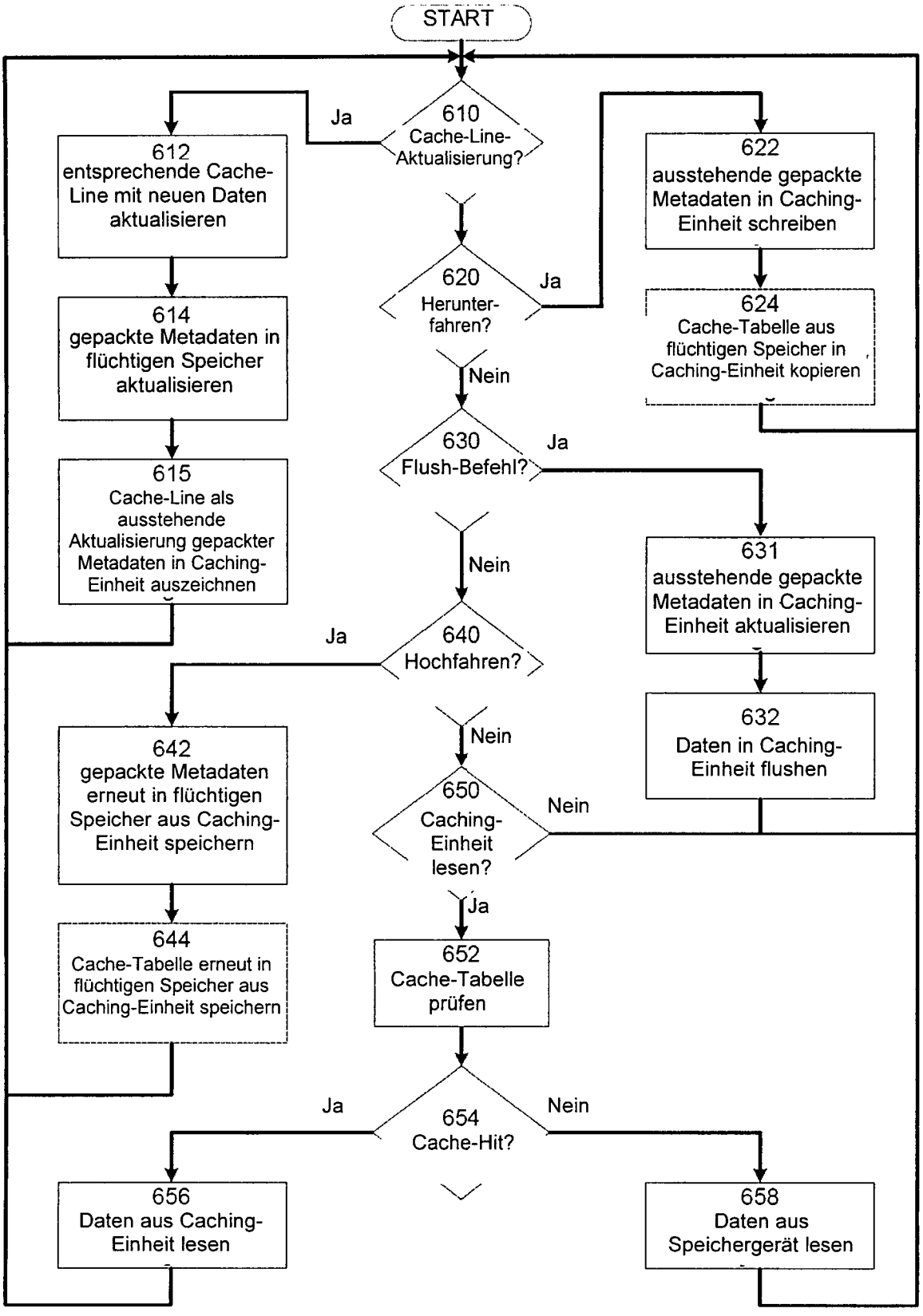

START
610
Cache-Line-Aktualisierung?
Ja
612
entsprechende Cache-Line mit neuen Daten aktualisieren
614
gepackte Metadaten in flüchtigen Speicher aktualisieren
615
Cache-Line als ausstehende Aktualisierung gepackter Metadaten in Caching-Einheit auszeichnen
620
Herunter-fahren?
Ja
622
ausstehende gepackte Metadaten in Caching-Einheit schreiben
624
Cache-Tabelle aus flüchtigen Speicher in Caching-Einheit kopieren
Nein
630
Flush-Befehl?
Ja
631
ausstehende gepackte Metadaten in Caching-Einheit aktualisieren
632
Daten in Caching-Einheit flushen
Nein
640
Hochfahren?
Ja
642
gepackte Metadaten erneut in flüchtigen Speicher aus Caching-Einheit speichern
644
Cache-Tabelle erneut in flüchtigen Speicher aus Caching-Einheit speichern
Nein
650
Caching-Einheit lesen?
Nein
Ja
652
Cache-Tabelle prüfen
654
Cache-Hit?
Ja
Nein
656
Daten aus Caching-Einheit lesen
658
Daten aus Speichergerät lesen
660

FIG. 6B

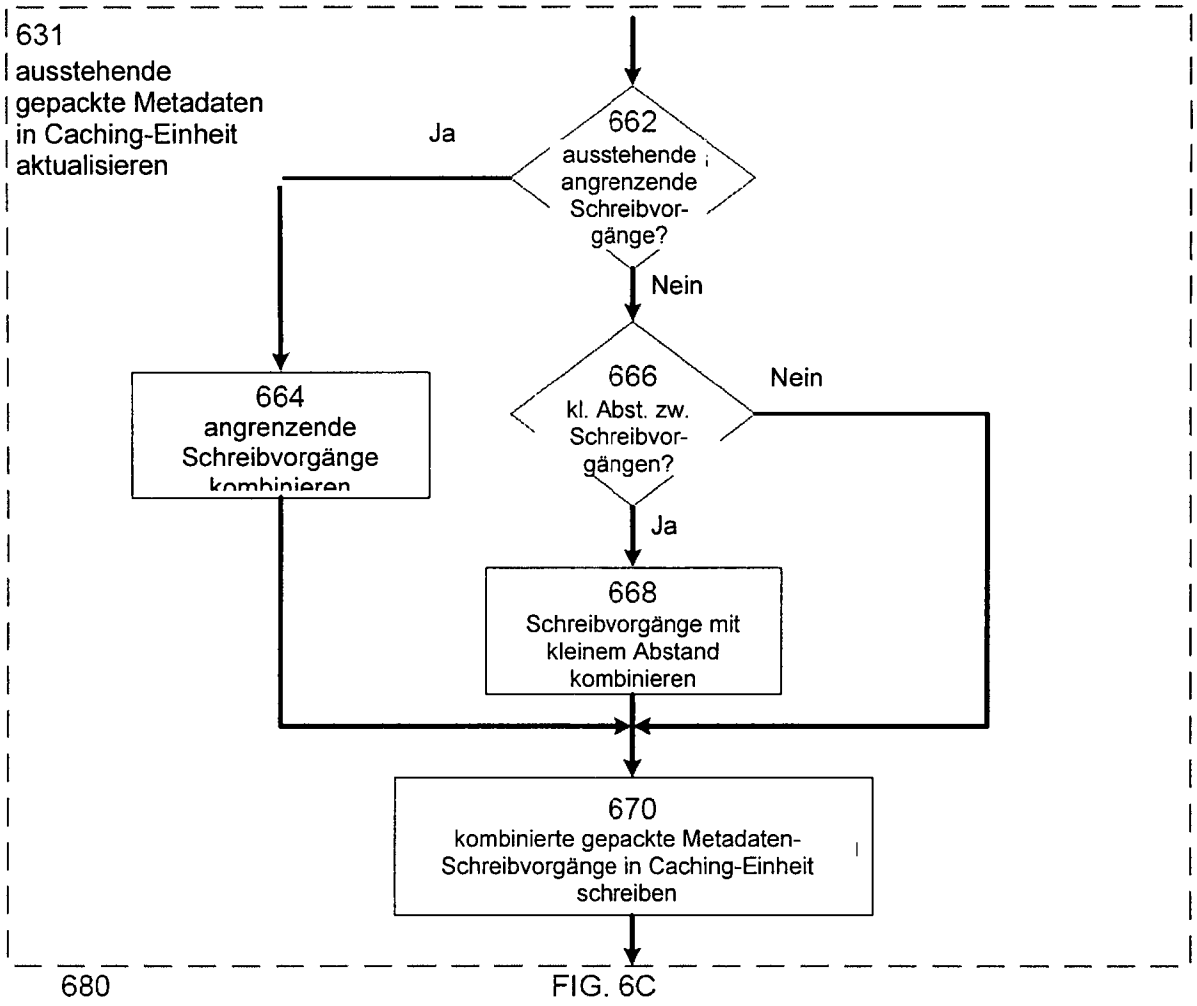
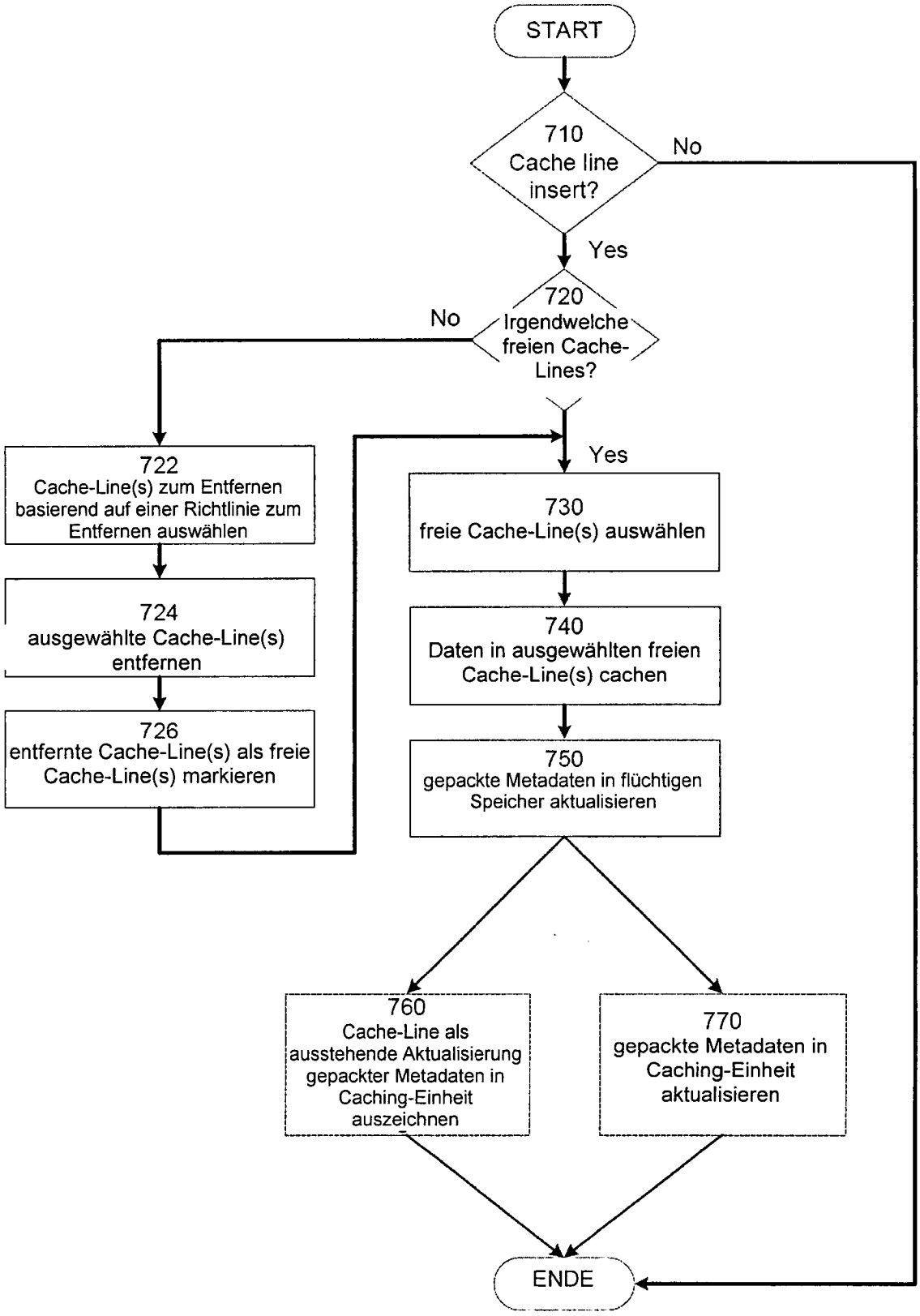

631
ausstehende gepackte Metadaten in Caching-Einheit aktualisieren
662
ausstehende angrenzende Schreibvorgänge?
Ja
Nein
664
angrenzende Schreibvorgänge kombinieren
666
kl. Abst. zw. Schreibvorgängen?
Nein
Ja
668
Schreibvorgänge mit kleinem Abstand kombinieren
670
kombinierte gepackte Metadaten-Schreibvorgänge in Caching-Einheit schreiben
680

FIG. 6C


START
710
Cache line insert?
No
Yes
720
Irgendwelche freien Cache-Lines?
No
Yes
722
Cache-Line(s) zum Entfernen basierend auf einer Richtlinie zum Entfernen auswählen
724
ausgewählte Cache-Line(s) entfernen
726
entfernte Cache-Line(s) als freie Cache-Line(s) markieren
730
freie Cache-Line(s) auswählen
740
Daten in ausgewählten freien Cache-Line(s) cachen
750
gepackte Metadaten in flüchtigen Speicher aktualisieren
760
Cache-Line als ausstehende Aktualisierung gepackter Metadaten in Caching-Einheit auszeichnen
770
gepackte Metadaten in Caching-Einheit aktualisieren
ENDE

700 FIG. 7

```
// Flüchtige Strukturen, die verwaltet werden müssen
SPARES = Liste von Cache-Lines, die keine Daten halten
LBA2CL = Cache-Tabelle. Ordnet LBA einer Cache-Line zu. LBA2CL[x] ist die Cache-Line, die die
Daten für LBA x hält, oder ist ansonsten -1.
META = enthält Metadaten für alle Cache-Lines. META[cl] sind die der Cache-Line cl
entsprechenden Metadaten. META[cl] enthält die LBA-Nummer des gecachten HDD-Sektors und
kann weitere Informationen enthalten.
PENDING_META_WRITES = Liste von Cache-Line-Nummern, für die Metadaten nicht flüchtig
geschrieben wurden.

// Layout der Daten der nicht flüchtigen Medien
NV_META = nicht flüchtige Metadaten. NV_META[cl] enthält Metadaten für Cache-Line cl.
NV_CACHE = Cache-Daten. NV_CACHE[cl] enthält die in Cache-Line cl gespeicherten
Benutzerdaten.

// InitializeNvStructures (wird nur ausgeführt, wenn Cache initialisiert oder zurückgesetzt
wird)
For cl = 0 to (NumCachelines – 1)
        nvmeta[cl].LBA = -1
End for
schreibe nvmeta nach NV_META

// InitializeVolatileStructures
// Initialisiert die flüchtigen Strukturen aus den nicht flüchtig gespeicherten Informationen.
// wird bei jeder Inbetriebnahme (sachgemäß oder unsachgemäß) durchgeführt
SPARES = {}
PENDING_META_WRITES = {}
lese nvmeta von NV_META
For cl = 0 to NumCachelines – 1
        lba = nvmeta[cl].lba
        META[cl] = nvmeta[cl]
        If (lba >= 0)
                LBA2CL[lba] = cl;
        Else
                SPARES = SPARES union { cl }
        End if
End for

// ProperShutdown
Cache-Flush
```

800 FIG. 8A

```
// Insert (LBA L, Data D, Metadata M)
// Fügt die spezifizierten Daten und Metadaten in den Cache ein.
// Die Metadaten bleiben flüchtig bis ein anschließender Flush ausgeführt wird.
If SPARES ist leer then
        lbaList = PickLBAsToEvict ()    // führt die Richtlinie zum Entfernen des Cache aus.
        Evict(lbaList)                  // kann dies mehrmals aufrufen, um mehrere //
                                        Cache-Lines zu entfernen. Dies fügt SPARES ein
                                        Element hinzu.
Endif cl = SPARES.RemoveElement ()
schreibe D nach NV_CACHE[cl]
PENDING_META_WRITES = PENDING_META_WRITES union { (cl,L) }
META[cl] = M
LBA2CL[L] = cl

// Evict (lbaList)
// Entfernt die spezifizierten Cache-Lines aus dem Cache.
For each L in lbaList
        cl = LBA2CL[L]
        Cache-Line cl säubern, wenn sie ungültig ist (d. h. wenn nicht sauber, Daten aus dem Cache
lesen und auf Disk schreiben).
        META[cl] = zeroes
        LBA2CL[L] = -1
        SPARES = SPARES union { cl }
        PENDING_META_WRITES = PENDING_META_WRITES union { (cl,L) }
End for UpdateNvMetadata ()

// ReadFromCache (LBA L)
// liest Daten aus dem Cache oder liefert einen Fehler zurück, wenn es ein Miss ist
cl = LBA2CL[L]
if (cl < 0)
        return NOT_IN_CACHE
end if
Lese SSD-Daten von NV_CACHE[cl] und liefere sie zurück.

// CacheFlush
// stellt sicher, dass alle Daten und Metadaten in dem Cache nicht flüchtig sind
UpdateNvMetadata ()
Flush an das SSD ausgeben

// UpdateNvMeta
// schreibt jegliche ausstehende Metadaten-Schreibvorgänge in den Cache
For each cl in PENDING_META_WRITES
        X = SSD-Sektor, der NV_META[cl] enthält
        SSD-Sektor X mit Informationen in META[cl] aktualisieren
End for
```

850

FIG. 8B