(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 11 2008 000 180 T5** 2009.12.03

(12) **Veröffentlichung**

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2008/087634**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2008 000 180.4**
(86) PCT-Aktenzeichen: **PCT/IL2008/000062**
(86) PCT-Anmeldetag: **16.01.2008**
(87) PCT-Veröffentlichungstag: **24.07.2008**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **03.12.2009**

(51) Int Cl.[8]: ***G06F 12/02*** (2006.01)
G11C 16/10 (2006.01)

(30) Unionspriorität:

60/885,412	**18.01.2007**	**US**
11/808,451	**11.06.2007**	**US**
11/808,452	**11.06.2007**	**US**

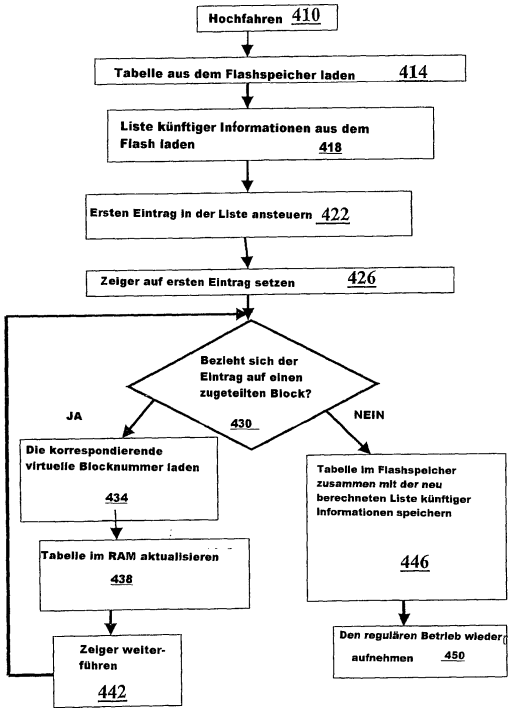
(71) Anmelder:
SanDisk IL Ltd., Kfar Saba, IL

(74) Vertreter:
Richardt, M., Dipl.-Ing., Pat.-Anw., 65343 Eltville

(72) Erfinder:
Lasser, Menahem, Kochav Yair, IL; Meir, Avraham, Rishon Le Zion, IL

(54) Bezeichnung: **Verfahren und System für die Umsetzung eines Fast-Wakeup eines Flashspeichersystems**

(57) Hauptanspruch: Speichermodul, gekennzeichnet durch:
(a) einen ersten nicht volatilen Speicher; und
(b) eine Steuerung des genannten ersten nicht volatilen Speichers, die dazu dient, den ersten nicht volatilen Speicher durch Schritte zu steuern, die folgende Schritte umfassen
(i) Speichern im genannten ersten nicht volatilen Speicher mindestens eines Teils einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt abbilden,
(ii) Speichern im genannten ersten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen in Bezug auf Ereignisse im Speichersystem umfasst, deren Eintritt nach dem Speichern der genannten künftigen Informationsdatenstruktur erwartet wird, und
(iii) zu einem zweiten Zeitpunkt, der auf das Speichern der genannten künftigen Informationsdatenstruktur folgt, Bearbeitung des genannten Ereignisses gemäß der künftigen Informationsdatenstruktur.



Beschreibung

ALLGEMEINER STAND DER TECHNIK

[0001] Die vorliegende Erfindung betrifft Verfahren und Systeme für die Vorhaltung von Datenstrukturen, die für die Umsetzung einer Wakeup-Funktion in einem Flash-Speichersystem nützlich sind.

[0002] Das US-Patent Nr. 6,510,488 von Lasser mit dem Titel „Method for a Fast Wake-Up of a Flash Memory System" (nachfolgend "Lasser '488") offenbart ein Verfahren und System, mit denen ein Flashspeichersystem ein schnelles Hochfahren nach Einschalten des Flash-Speichersystems erreicht, selbst wenn die Flashsystemsoftware Managementtabellen verwendet, deren Neuerstellung zeitaufwändig ist. Die kurze Wakeup-Zeit wird ohne Beeinträchtigung der Datenintegrität erreicht. Das genannte Patent von Lasser '488 ist vorliegend für alle Zwecke durch Hinweis so aufgenommen als sei es in vorliegender Patentschrift vollumfänglich enthalten.

[0003] Wie in Lasser '488 dargelegt, erfordert die Verwendung einer Flashspeichervorrichtung für Computerdaten traditionell eine Software-Translationsschicht zwischen dem Betriebssystem des Hauptrechners und den systemnahen Zugangsroutinen der Vorrichtung. Dies ist der Fall, da die Flashtechnologie einige Nutzungsbeschränkungen aufweist, die es unmöglich machen, zum Flashspeicher einfach durch direkten linearen Zugriff Zugang zu erlangen. Eine dieser Beschränkungen ist die Unmöglichkeit, einen beliebigen Ort im Flashspeicher direkt zu überschreiben. Genau gesagt, kann das Schreiben von neuem Inhalt in einen Flashspeicher zuerst das Löschen des gesamten Blocks, in dem der betreffende Ort sich befindet, erfordern (unter Erhaltung der Inhalte in anderen noch benötigten Orten), bevor dann der neue Inhalt eingegeben werden kann.

[0004] Die Translationsschicht präsentiert dem Betriebssystem des Hauptrechners eine virtuelle Ansicht einer Reihe unabhängiger und direkt zugänglicher Datensektoren und verbirgt zugleich die Einzelheiten des Mapping dieser virtuellen Adressen zu deren realen Orten im Flashmedium. Dieser Translationsmechanismus ist alles andere als trivial, und ein Beispiel einer solchen Translationsschicht für einen Flashspeicher ist in Amir Bans US-Patent Nr. 5,937,425 offenbart, das vorliegend durch Bezugnahme aufgenommen ist. Ban offenbart ein Verfahren für die Umsetzung eines Mappingmechanismus zwischen virtuellen und physischen Flashadressen. Ein weiteres Beispiel eines solchen Systems ist in US-Patent Nr. 6,678,785 offenbart, das ebenfalls durch Hinweis vorliegend aufgenommen ist.

[0005] Der Translationsprozess nutzt interne Translationstabellen, die der Flashsystemsoftware die für die Konversion der Zugangsanfragen vom Hauptrechner in Zugangsanfragen zum Flashgerät erforderlichen Informationen liefern. Das Flashspeichersystem erstellt diese Translationstabellen während des Wakeup (oder später, sofern von der Betriebssoftware des Hauptrechners so vorgegeben) auf Grundlage der im Flashgerät gespeicherten Steuerinformationen. Theoretisch ist es zwar möglich, solche Tabellen nicht zu erstellen und nur die rohen Steuerdaten des Flashspeichers zu nutzen. In der Praxis ist dies jedoch nicht möglich, da die Reaktionszeit auf eine Zugangsanfrage zu langsam wäre. Das ist der Fall, da der Zugriff auf Daten auf einem Flashgerät viel langsamer erfolgt als der Datenzugriff in einem RAM-Speicher, und auch da die RAM-Speichertabellen für während der Laufzeit erforderliche Operationen gewöhnlich in Hinblick auf ihre Effizienz optimiert werden, was bei Steuerdaten in Flashspeichern nicht der Fall ist.

[0006] So kann beispielsweise eine physische Flasheinheit die Zahl der auf sie zugreifenden virtuellen Einheiten enthalten. Während der Laufzeit des Programms müssen wir unter Umständen häufig die virtuelle Nummer einer Einheit in ihr physisches Äquivalent konvertieren. Müssen wir uns dabei ausschließlich auf die im Flashspeicher hinterlegten Steuerdaten verlassen, müssen wir möglicherweise die Einheiten abtasten, bis wir die Einheit mit der angegebenen Nummer der virtuellen Einheit finden. Das ist für einen einfachen Medienzugang ein sehr langer Prozess. Wird jedoch beim Hochfahren des Systems der Flashspeicher einmal abgetastet und eine Tabelle erstellt, die jede virtuelle Einheitsnummer der Nummer der entsprechenden physischen Einheit zuordnet, kann dieses Mapping später sehr effizient durchgeführt werden.

[0007] Das Problem besteht darin, dass das Scannen des Flashdatenspeichergeräts beim Hochfahren des Systems lange dauern kann, insbesondere bei Geräten hoher Speicherkapazität. Das ist besonders ärgerlich bei Systemen und Geräten, deren Anwender eine umgehende Betriebsbereitschaft erwarten (z. B. Mobiltelefonen, PDA etc.). Bei Read-Only-Geräten wie Flashgeräten, die nur Computercode speichern, der vom Anwender nicht änderbar ist, mag einfaches Speichern der Tabellen im Flashspeicher ausreichen. Doch das bloße Speichern der Tabellen im Flashspeicher ist bei Geräten, die verwendet werden, um Daten zu speichern, die sich häufig ändern können (z. B. Textdateien oder Spreadsheets in einem PDA), nicht erfolgreich. Denn wenn laufende Eingaben in das Gerät erfolgen und die Inhalte im Geräte laufend geändert werden, ändern sich die Inhalte der Translationstabellen ebenfalls. Es ist nicht praktikabel, die Kopie der Tabellen im Flashspeicher jedes Mal zu hochfahren, wenn sich die Tabellen im RAM ändern, denn die dadurch entstehende Arbeitslast wird das System deutlich verlangsamen. Folglich

akkumuliert sich eine Abweichung zwischen den im Flashspeicher gespeicherten Tabellen und den „richtigen" im RAM. Schaltet der Anwender nun die Stromversorgung ab und dann wieder an ohne die Tabellen zu hochfahren, wird die Software die fehlerhaften Translationstabellen aus dem Flashspeicher lesen, mit der Folge eines möglichen Datenverlusts bei der Eingabe neuer Daten.

[0008] Gemäß manchen in Lasser '488 offenbarten Ausführungsbeispielen wird das Problem gelöst, indem die Translationstabellen im Flashspeicher gespeichert werden und Mittel hinzugefügt werden, die es der Software ermöglichen, die Translationstabellen auf eine Weise zu invalidieren, die jedes Mal feststellbar ist, wenn sie gelesen werden. Zu den möglichen Umsetzungen (doch nicht den einzigen) zählt die Hinzufügung eines Prüfsummenwerts, der die Summe aller Eingaben einem festen bekannten Wert gleichsetzt, oder das Hinzufügen einer Wirksamkeitsanzeige zu den gespeicherten Tabellen. Außerdem sollte man die Anwendungssoftware auffordern, eine bestimmte Funktion in der Translationsschicht aufzurufen, bevor das System ausgeschaltet wird.

[0009] Mittels dieser Maßnahmen kann das Flashspeichergerät schnelles Hochfahren einleiten, wenn das System ordnungsgemäß heruntergefahren wurde, und sich auf das reguläre Hochfahren umstellen, wenn es nicht ordnungsgemäß abgeschaltet wird.

[0010] Diese Lösung ist zwar in vielen Fällen nützlich, doch es gibt Situationen, in denen sie möglicherweise nicht ausreicht. Ein erstes Beispiel, in denen sie nicht ausreichend sein könnte, sind häufige Stromausfälle, so dass zu erwarten ist, das viele (oder sogar die meisten) der Einschaltereignisse auf unwirksame gespeicherte Tabellen stoßen, so dass es zu langsamem regulären Hochfahren kommt.

[0011] Ein zweites Beispiel, in dem die Lösung ungenügend sein kann, ist wenn das Betriebssystem des Geräts, auf dem der Flashspeicher gehostet wird, der Softwareanwendung keine Funktion für die ordnungsgemäße Demontage oder das ordnungsgemäße Abschalten anbietet. Während komplexe Betriebssysteme wie Linux solche Funktionen anbieten, gibt es viele einfachere und kleinere Betriebssysteme, die darauf ausgelegt sind, das Speichersystem bei Einschalten der Stromversorgung hochzufahren, und nie das Betriebssystem ausschalten. In solchen Fällen führen die Methoden gemäß Lasser '488 dazu, dass jedes Einschalten des Geräts ein reguläres Hochfahren des Flashmanagementsystems auslöst, so dass mit diesen Methoden nichts gewonnen ist.

[0012] Ein drittes Beispiel, in dem die Lösung ungenügend sein kann, ist der Fall, dass das Zeitintervall zwischen dem Einschalten des Systems und der Betriebsbereitschaft des Systems streng begrenzt ist. Selbst wenn Stromausfälle selten sind und fast jedes Einschalten zu einem raschen Hochfahren des Flashmanagementsystems führt, ist es nicht akzeptabel, dass ein Stromausfall zu einer späteren regulären Hochfahrsequenz führt, wie selten dies auch immer der Fall sein mag.

[0013] In Anbetracht der vorstehend beschriebenen Mängel von Lasser '488 offenbart die US-Patentanmeldung 11/382,056 Lasser (nachfolgend „Lasser '056") eine weitere Lösung des Problems des raschen Hochfahrens von Flashmanagementsystemen. Die genannte Anmeldung Lasser '056 wird vorliegend vollständig einbezogen.

[0014] Lasser '056 offenbart eine Technik, mit der eine oder mehrere Flashmanagementtabellen nach manchen, aber nicht allen Ereignissen im Flashspeichersystem aktualisiert und gespeichert werden. Stellt sich beim Hochfahren heraus, dass eine gegebene im Flashspeicher gespeicherte Flashmanagementtabelle veraltete Informationen enthält, lassen sich die gespeicherte Tabelle(n) dennoch für das Hochfahren des Systems verwenden, und es ist nicht erforderlich, die veraltete Tabelle zu invalidieren. Statt sie zu invalidieren, kann die im Flashspeicher vor dem Abschalten und/oder Stromausfall gespeicherte veraltete Flashmanagementtabelle beim Hochfahren verwendet werden, um die „richtige" Tabelle zu rekonstruieren (d. h. die, die dem aktuellen Status des Systems entspricht).

[0015] Dies erfolgt, indem im Flashspeicher ein Ereignisprotokoll geführt wird. Beim Hochfahren werden im Ereignisprotokoll hinterlegte Daten verwendet, um die Flashspeichertabelle zu aktualisieren und somit die Datenintegrität zu sichern, selbst wenn das System vor dem Ausschalten oder Stromausfall nicht ordnungsgemäß heruntergefahren wurde. In den meisten Fällen erfolgt der Abruf einer „aktualisierten" aus einer im Flashspeicher hinterlegten „veralteten" Tabelle unter Zugriff auf ein Ereignisprotokoll schneller als die Konstruktion einer aktualisierten Tabelle mittels Scannen des Flashspeichers.

[0016] Ein Nachteil von Lasser '056 besteht darin, dass ein Ereignisprotokoll im Flashspeicher vorgehalten werden muss. Das ist zwar in manchen Flashmanagementsystemen kein großer Mangel, da in ihnen schon aus anderen Gründen ein Ereignisprotokoll geführt wird. Doch es gibt viele Flashmanagementsysteme, in denen ein Ereignisprotokoll ansonsten nicht benötigt wird, so dass die Verfahren gemäß Lasser '056 in Hinblick auf die Schreibleistung aufwändig sind.

[0017] Folglich besteht ein allgemein anerkannter Bedarf an einem Verfahren und System, das eine Methode für das rasche Hochfahren eines Flashspeichersystems bietet, ohne die Integrität der Flash-Da-

tenstrukturen oder die Systemleistung zu beeinträchtigen, und es wäre von Vorteil, über ein solches Verfahren und System zu verfügen.

DEFINITIONEN

[0018] Für die Zwecke der vorliegenden Offenlegungsschrift bezeichnet der Ausdruck „Block" die kleinste Einheit des Flashspeichers, die in einem einzigen Vorgang gelöscht werden kann. Der Ausdruck „Seite" bezeichnet die kleinste Einheit des Flashspeichers, die in einem einzigen Vorgang geschrieben werden kann (aus historischen Gründen auch „programmiert" genannt). Ein Block umfasst im Allgemeinen viele Seiten.

[0019] Für die Zwecke der vorliegenden Offenlegungsschrift sind die Ausdrücke „Flashmanagementsystem" und „Flashdateisystem" synonym und werden austauschbar verwendet. Beide Begriffe bezeichnen ein Softwaremodul, das die Speicherung von Daten in einem Flashspeichergerät steuert, unabhängig davon, ob die vom Modul exportierte Schnittstelle dateiorientiert ist (mit Befehlen wie „Datei öffnen" oder „Datei schreiben") oder blockorientiert (mit Befehlen wie „Block lesen" oder „Block schreiben") und ungeachtet der Tatsache, ob das Modul auf einem Steuergerät läuft, das ausschließlich dem Flashmanagement dient, oder auf dem gleichen Hauptrechner, auf dem auch die Anwendungen laufen, die das Speichersystem nutzen.

[0020] Für die Zwecke der vorliegenden Offenlegungsschrift bezeichnet „Flashmanagementtabelle" jede Tabelle, die Daten enthält, die von einem Flashmanagementsystem zur Unterstützung des Betriebs seiner Algorithmen verwendet werden, wobei die Daten in der Tabelle zu jedem gegebenen Zeitpunkt einen Aspekt des Status des Flashspeichersystems zu diesem spezifischen Zeitpunkt abbilden. Ist die Flashmanagementtabelle zum Beispiel eine Tabelle, die ein Bit für jeden Block des Flashspeichers enthält, wobei das Bit anzeigt, ob der entsprechende Block für die Verwendung frei zur Verfügung steht, dann sind die Inhalte der Tabelle zu einem ersten Zeitpunkt ein erstes Bitmuster, das den Aspekt des Systemstatus anzeigt, welche der Blöcke zu diesem Zeitpunkt frei und welche nicht frei sind. Zu einem späteren Zeitpunkt könnte das Bitmuster in der Tabelle demjenigen zum ersten Zeitpunkt entsprechen oder sich von demjenigen zum ersten Zeitpunkt unterscheiden, was eine unterschiedliche Kombination freier und nicht freier Blöcke impliziert, die dadurch verursacht worden ist, dass manche freie Blöcke nun nicht frei sind und andere zuvor nicht freie frei geworden sind.

[0021] Für die Zwecke der vorliegenden Offenlegungsschrift bezeichnet „Ereignis" jeden Schreibbefehl, Löschbefehl oder Verwaltungsbefehl an den Flashspeicher von einer Einheit, die den Flashspeicher steuert. Die Einheit könnte eine Flashspeichersteuerung gemäß nachstehender **Abb. 1**, eine Standard-CPU gemäß nachstehender **Abb. 5** oder eine Flashspeichersteuerung und Standard-CPU, die zusammenwirken, gemäß nachstehender **Abb. 6** sein.

[0022] Für die Zwecke der vorliegenden Offenlegungsschrift bezeichnet „ausgewähltes Ereignis" ein vom Systementwickler des Flashspeichersystems ausgewähltes Ereignis, das die Erstellung einer oder mehrerer Aktualisierungen einer oder mehrerer Speichermanagementtabellen anstößt.

BESCHREIBUNG DER ERFINDUNG

[0023] Einige oder alle vorgenannten Anforderungen sowie weitere Anforderungen werden durch mehrere Aspekte der vorliegenden Erfindung erfüllt.

[0024] Die vorliegende Erfindung umfasst ein Verfahren, das zur Erfüllung der genannten Anforderung im Flashspeicher Informationen zu künftigen Ereignissen im Flashspeichersystem hinterlegt. Das System speichert zum Beispiel während der Speicherung des aktuellen Werts in einer Translationstabelle für die Konvertierung virtueller in physische Adressen gleichzeitig mit der Tabelle auch eine Liste der nächsten physischen Blöcke, die künftig zu verwenden sind, wenn Ereignisse im System die Zuteilung neuer freier physischer Blöcke erfordern.

[0025] Zu einem späteren Zeitpunkt kann ein Ereignis im System (z. B. ein Schreibbefehl) die Zuteilung eines neuen physischen Blocks auslösen, der einen anderen physischen Block als den korrespondierenden Block eines virtuellen Blocks ersetzen soll, wodurch eine Änderung im Status der die Adressen mappierenden Flashmanagementtabelle ausgelöst wird. Tritt ein solches Ereignis ein, wird die Flashmanagementsoftware als neu zugeteilten Block den Block an erster Stelle in der gespeicherten Liste zuteilen. Nach Abschluss der Bearbeitung des Schreibbefehls ist die gespeicherte Kopie der Tabelle nicht mehr aktuell, da das durch die Schreiboperation erfolgte Mappieren des virtuellen Blocks nicht mehr dem aktuellen Zustand des Systems entspricht.

[0026] Wird das System hochgefahren, wird zuerst die gespeicherte Kopie der Managementtabelle geladen. Zu diesem Zeitpunkt besteht keine Gewissheit, dass die Tabelle den Status des Systems richtig abbildet, da die Möglichkeit besteht, dass ein paar Ereignisse vor Abschalten des Systems mehrere Inkompatibilitäten zwischen der hinterlegten Tabelle und der richtigen Tabelle verursacht haben. Doch die im Flashspeicher hinterlegte und die hinterlegte Tabelle betreffende „künftige Information" bietet genügend Angaben für die Korrektur der hinterlegten Tabelle und Rekonstruktion der richtigen aktualisierten Tabellenversion.

[0027] Dies ist möglich, da wir wissen, dass jede Zuteilung physischer Blocks, die den Mappingstatus geändert hat (soweit einer bestand), die in der Liste aufgeführten physischen Blöcke verwendet haben muss. Außerdem müssen die Blöcke exakt in der in der Liste vorgegebenen Reihenfolge verbraucht worden sein. Um zu bestimmen, ob die gespeicherte Tabelle aktuell ist oder nicht, ist es folglich ausreichend, den Systemstatus in Hinblick auf den physischen Block, der an erster Stelle in der Liste steht, zu prüfen. Ist dieser erste Block noch immer frei, ist keine neue Zuteilung erfolgt, und die Tabelle ist aktuell. Ist der erste Block in der Liste hingegen nicht mehr frei, muss eine Änderung im System eingetreten sein, und die Tabelle ist nicht mehr aktuell. In diesem Fall finden wir die Aktualisierung, die in der Tabelle vorzunehmen ist, um der Zuteilung des ersten Blocks in der Liste zu entsprechen, und aktualisieren die Kopie der Tabelle im RAM. Stellt sich heraus, dass der erste Block verwendet wird, müssen wir nacheinander den gleichen Test für die nächsten Blöcke in der Liste durchführen und die gleiche Logik wiederholt anwenden. Dies wird wiederholt, bis wir in der Liste einem Block begegnen, der noch nicht verwendet worden ist, oder bis wir zum Ende der Liste gelangen.

[0028] Um festzustellen, ob ein Block zurzeit verwendet wird, und die Änderungen zu bestimmen, die an der Tabelle vorzunehmen sind, sofern der Block zurzeit verwendet wird, wird Zugang zu den Steuerdaten im Flashspeicher und möglicherweise im betreffenden Block selbst benötigt. Obgleich der Flashspeicherzugang im Vergleich zum RAM-Zugang langsam ist, benötigt das Verfahren gemäß vorliegender Erfindung nur eine geringe Anzahl an Zugängen zum Flashspeicher, da nur Blöcke in der „Zukunftsliste" geprüft werden und generell nur einige Blöcke gemäß der vorstehend beschriebenen Logik zu prüfen sind. Daher ist das vorliegende Verfahren für die Generierung einer aktualisierten Version eines Flashmanagementsystems aus einer veralteten Version der Tabelle sehr viel schneller als eine vollständige Rekonstruktion der Tabelle mittels umfassenden Scannens aller Blöcke des Flashspeichers.

[0029] Die vorliegende Erfindung lässt sich in gewisser Weise als Analogie zum Verfahren gemäß Lasser '056 betrachten. Beide Verfahren rekonstruieren eine aktualisierte Version einer Managementtabelle aus einer gespeicherten Kopie der Tabelle mit Hilfe zusätzlicher flashgespeicherter Informationen. Doch während in Lasser '056 die zusätzlichen Daten ein Ereignisprotokoll sind, sind sie in vorliegender Erfindung eine auf „künftige Ereignisse" bezogene Liste. In Lasser '056 werden die zusätzlichen Informationen zu einem Zeitpunkt hinterlegt, der später ist als der Zeitpunkt des Speicherns der Tabelle im Flashspeicher, während in vorliegender Erfindung die zusätzlichen Informationen generell zum gleichen Zeitpunkt gespeichert werden, an dem auch die Tabelle im Flashspeicher hinterlegt wird.

[0030] Hiermit wird erstmals ein Verfahren für die Pflege der Datenstrukturen eines Speichersystems auf Grundlage der Ereignisse im System offen gelegt, das die folgenden Schritte umfasst: (a) Speicherung mindestens eines Teils einer Managementtabelle, deren Inhalte einen Status des Speichersystems anzeigen, in einem nicht volatilen Speicher des Speichersystems zu einem ersten Zeitpunkt; (b) Speicherung einer künftigen Informationsdatenstruktur einschließlich einer Vielzahl von Datensätzen, die sich auf Ereignisse im Speichersystem beziehen, deren Eintritt nach Speicherung der Informationsdatenstruktur erwartet wird, im nicht volatilen Speicher; und (c) zu einem zweiten Zeitpunkt, der zeitlich nach dem ersten Zeitpunkt folgt, Bearbeitung eines Ereignisses gemäß der künftigen Informationsdatenstruktur.

[0031] Der nicht volatile Speicher ist vorzugsweise ein Flashspeicher. Gemäß einigen Ausführungsbeispielen umfasst die Speicherung von mindestens einem Teil mindestens einer Managementtabelle die Aktualisierung von mindestens einem Teil mindestens einer Managementtabelle im nicht volatilen Speicher.

[0032] Gemäß manchen Ausführungsbeispielen umfasst die Speicherung von mindestens einem Teil mindestens einer Managementtabelle, deren Inhalte den Status des Speichersystems anzeigen, auch die Speicherung mindestens eines Teils mindestens einer Managementtabelle in einem volatilen Speicher des Speichersystems. Die Aktualisierung erfolgt alle $N > 1$ Male, wenn mindestens ein Teil von mindestens einer Managementtabelle im volatilen Speicher geändert wird. Andere Arten der Durchführung der Aktualisierung als Reaktion auf ausgewählte Ereignisse umfassen die periodische Aktualisierung und die Aktualisierung entsprechend der Verfügbarkeit von Kapazität im Speichersystem.

[0033] Gemäß einer Reihe von Ausführungsbeispielen umfasst die Datenstruktur für künftige Informationen eine Liste von Blöcken im nicht volatilen Speicher, die als erste frei sind.

[0034] Vorliegend wird erstmals ein Verfahren für die Aktivierung eines Speichersystems offen gelegt, das folgende Schritte umfasst: (a) das Lesen mindestens eines Teils mindestens einer Managementtabelle, die einen Status des Speichersystems zu einem Zeitpunkt vor dem Hochfahren des Systems beschreibt, aus einem nicht volatilen Speicher des Speichersystems; (b) das Lesen einer künftigen Informationsdatenstruktur, die eine Vielzahl von Datensätzen in Bezug auf Ereignisse enthält, deren Eintritt nach Speicherung der künftigen Informationsdatenstruktur erwartet wird, aus dem nicht volatilen Speicher; und (c) die Aktualisierung mindestens eines Teils mindes-

tens einer nicht volatilen Managementtabelle gemäß mindestens einem Datensatz der künftigen Informationsdatenstruktur.

[0035] Gemäß manchen Ausführungsbeispielen ist der nicht volatile Speicher ein Flashspeicher.

[0036] Gemäß manchen Ausführungsbeispielen verändert das Aktualisieren mindestens einen Teil mindestens einer Managementtabelle, um einen aktuellen Status des Speichersystems abzubilden. Die Aktualisierung erfolgt bedingt. Erfolgte der Systemausstieg ordnungsgemäß, ist keine Aktualisierung erforderlich. Gemäß manchen Ausführungsbeispielen umfasst das Verfahren beispielsweise auch den Schritt des Vergleichens einer Vielzahl von Datensätzen mit dem nicht volatilen Speicher, um festzustellen, ob sich der Status des Speichersystems seit jenem Zeitpunkt geändert hat, so dass die Aktualisierung dann davon abhängig erfolgt, ob sich der Status des Speichersystems seit diesem Zeitpunkt verändert hat.

[0037] Vorliegend wird erstmals ein Speichermodul offen gelegt, das (a) einen ersten nicht volatilen Speicher; und (b) eine Steuerung des nicht volatilen Speichers umfasst, der dazu dient, den ersten nicht volatilen Speicher mittels folgender Schritte zu steuern: (i) Speichern im ersten nicht volatilen Speicher mindestens eines Teils mindestens einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt anzeigen; (ii) Speichern im ersten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, einschließlich einer Vielzahl von Datensätzen in Bezug auf Ereignisse im Speichersystem, deren Eintritt nach Speicherung der künftigen Informationsdatenstruktur erwartet wird; und (iii) zu einem zweiten Zeitpunkt nach Speicherung der künftigen Informationsdatenstruktur Bearbeitung des Ereignisses gemäß der künftigen Informationsdatenstruktur.

[0038] Gemäß manchen Ausführungsbeispielen umfasst das Modul des Weiteren einen zweiten nicht volatilen Speicher; und die Steuerung dient zur Umsetzung der Schritte mittels Ausführung eines im zweiten nicht volatilen Speicher hinterlegten Codes.

[0039] Vorliegend wird erstmals ein Speichersystem offen gelegt, das Folgendes umfasst: (a) ein Speichermodul, das einen nicht volatilen Speicher enthält; und (b) ein Hostrechner des Speichermoduls, der bei der Verwaltung des nicht volatilen Speichers u. a. durch folgende Schritte mitwirkt: (i) Speicherung mindestens eines Teils mindestens einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt anzeigen, im ersten nicht volatilen Speicher; (ii) Speicherung einer künftigen Informationsdatenstruktur, einschließlich einer Vielzahl von Datensätzen in Bezug auf Ereignisse im Speichersystem, deren Eintritt nach Hinterlegung der besagten Informationsdatenstruktur erwartet wird, in dem ersten nicht volatilen Speicher; und (iii) zu einem zweiten Zeitpunkt nach Speicherung der künftigen Informationsdatenstruktur Bearbeiten eines Ereignisses gemäß den Vorgaben der künftigen Informationsdatenstruktur.

[0040] Gemäß manchen Ausführungsbeispielen werden diese Schritte ausschließlich vom Hostrechner ausgeführt. Gemäß anderen Ausführungsbeispielen umfasst das Speichermodul auch eine Steuerung, die mit dem Hostrechner bei der Ausführung der Schritte zusammenwirkt.

[0041] Vorliegend wird erstmals ein Speichermodul offen gelegt, das Folgendes umfasst: (a) einen ersten nicht volatilen Speicher; und (b) eine Steuerung des nicht volatilen Speichers, der dazu dient, das Speichermodul mittels folgender Schritte zu aktivieren: (i) Auslesen aus dem ersten nicht volatilen Speicher mindestens eines Teils mindestens einer Managementtabelle, die einen Status des Speichermoduls zu einem Zeitpunkt vor der Aktivierung beschreibt; (ii) Auslesen aus dem nicht volatilen Speicher einer künftigen Informationsdatenstruktur einschließlich einer Vielzahl von Datensätzen in Bezug auf Ereignisse, deren Eintritt nach diesem Zeitpunkt erwartet wird; und (iii) Aktualisieren mindestens eines Teils der mindestens einen Flashmanagementtabelle gemäß mindestens eines Datensatzes der künftigen Informationsdatenstruktur.

[0042] Gemäß einigen Ausführungsbeispielen umfasst das Speichermodul des Weiteren einen zweiten nicht volatilen Speicher, und die Steuerung dient dazu, die Schritte durch Ausführung eines im zweiten nicht volatilen Speicher hinterlegten Codes umzusetzen.

[0043] Vorliegend wird erstmals ein Speichersystem offen gelegt, das Folgendes umfasst: (a) ein Speichermodul mit einem nicht volatilen Speicher; und (b) einen Hostrechner des Speichermoduls, der an der Verwaltung des nicht volatilen Speichers u. a. mittels folgender Schritte mitwirkt: (i) Auslesen aus dem nicht volatilen Speicher mindestens eines Teils mindestens einer Managementtabelle, die einen Zustand des Speichersystems zu einem Zeitpunkt vor der Aktivierung beschreibt; (ii) Auslesen aus dem nicht volatilen Speicher einer künftigen Informationsdatenstruktur, einschließlich einer Vielzahl von Datensätzen in Bezug auf Ereignisse, deren Eintritt nach Speicherung der künftigen Informationsdatenstruktur erwartet wird; und (iii) Aktualisierung der mindestens einen Managementtabelle gemäß mindestens einem Datensatz der künftigen Informationsdatenstruktur.

[0044] Gemäß manchen Ausführungsbeispielen werden die Schritte ausschließlich durch den Host-

rechner ausgeführt. Gemäß anderen Ausführungsbeispielen umfasst das Speichermodul eine Steuerung, die mit dem Hostrechner bei der Umsetzung der Schritte zusammenwirkt.

[0045] Vorliegend wird erstmals ein computerlesbares Speichermedium offen gelegt, das computerlesbaren Code für die Pflege der Datenstrukturen eines Speichersystems gemäß den Ereignissen im System umfasst, wobei der computerlesbare Code Folgendes umfasst: (a) Programmcode für die Speicherung mindestens eines Teils mindestens einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt anzeigen, in einem nicht volatilen Speicher des Speichersystems; (b) Programmcode für die Speicherung einer künftigen Informationsdatenstruktur einschließlich einer Vielzahl von Datensätzen in Bezug auf Ereignisse im Speichersystem, deren Eintritt nach Speicherung der künftigen Informationsdatenstruktur erwartet wird, im nicht volatilen Speicher; und (c) Programmcode für die Bearbeitung des Ereignisses gemäß der künftigen Informationsdatenstruktur zu einem zweiten Zeitpunkt nach Speicherung derselben.

[0046] Vorliegend wird erstmals ein computerlesbares Speichermedium offen gelegt, das computerlesbaren Code für die Aktivierung eines Speichersystems umfasst, wobei der computerlesbare Code Folgendes umfasst: (a) Programmcode für das Auslesen mindestens eines Teils mindestens einer Managementtabelle, die einen Status des Speichersystems zu einem Zeitpunkt vor der Aktivierung beschreibt, aus einem nicht volatilen Speicher des Speichersystems; (b) Programmcode für das Auslesen einer künftigen Informationsdatenstruktur, die eine Vielzahl von Datensätzen bezüglich Ereignissen enthält, deren Eintritt nach Speicherung der künftigen Informationsdatenstruktur erwartet wird, aus dem nicht volatilen Speicher; und (c) Programmcode für die Aktualisierung mindestens eines Teils der mindestens einen Managementtabelle gemäß mindestens einem Datensatz der künftigen Informationsdatenstruktur.

KURZBESCHREIBUNG DER ABBILDUNGEN

[0047] Abb. 1 ist ein Blockdiagramm eines Ausführungsbeispiels eines Flashspeichersystems gemäß mancher Ausführungsbeispiele der vorliegenden Erfindung;

[0048] Abb. 2A–Fig. 2B zeigen ein Beispiel einer Translationstabelle gemäß mancher Ausführungsbeispiele vorliegender Erfindung;

[0049] Abb. 3 ist ein Flussdiagramm der Pflege einer Flashmanagementtabelle in einem Flashspeicher sowie einer künftigen Informationsdatenstruktur gemäß eines Ausführungsbeispiels vorliegender Erfindung;

[0050] Abb. 4 ist ein Flussdiagramm einer beispielhaften Routine für die Aktivierung.

[0051] Abb. 5 und Abb. 5 sind Blockdiagramme weiterer beispielhafter Flashspeichersysteme gemäß mancher Ausführungsbeispiele vorliegender Erfindung.

BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSBEISPIELE

[0052] Die vorliegende Erfindung wird nun anhand spezifischer Ausführungsbeispiele beschrieben. Es sei darauf hingewiesen, dass die Erfindung sich nicht auf die offen gelegten Ausführungsbeispiele beschränkt. Es sei weiter darauf hingewiesen, dass nicht jedes Merkmal der vorliegend offen gelegten Verfahren, Vorrichtungen und computerlesbaren Codes für die Pflege von Datenstrukturen auf Grundlage der Ereignisse in einem Flashspeichersystem erforderlich ist, um die Erfindung gemäß einem der nachstehend beigefügten Ansprüche umzusetzen. Verschiedene Elemente und Merkmale von Vorrichtungen werden beschrieben, um die vollständige Umsetzung der Erfindung zu ermöglichen. Es sei außerdem darauf hingewiesen, dass die Schritte in vorliegender Offenlegungsschrift gezeigter oder beschriebener Verfahren oder Methoden in beliebiger Reihenfolge oder gleichzeitig durchgeführt werden können, es sei denn aus dem Zusammenhang ist klar, dass ein Schritt die vorherige Ausführung eines anderen Schrittes erfordert.

[0053] Die vorliegend offen gelegten Verfahren, Systeme und computerlesbaren Codes für die Pflege von Datenstrukturen sind nützlich für die Umsetzung eines „raschen Wakeup" des Flashspeichersystems, zum Beispiel in Umgebungen, in denen häufig Stromausfälle auftreten. Das ist jedoch nicht als Beschränkung vorliegender Erfindung auszulegen und wird lediglich als eine nicht beschränkende Anwendung der vorliegend offen gelegten Techniken für die Pflege der Datenstrukturen von Flashspeichersystemen offen gelegt.

[0054] Die vorliegend offen gelegten Techniken dienen der Umsetzung eines „raschen Wakeup" eines Flashmanagementsystems ohne Beeinträchtigung der Datenintegrität selbst unter Bedingungen, in denen häufig unvorhersehbare Stromausfälle auftreten.

[0055] Bezug nehmend auf die Zeichnungen ist Abb. 1 ein Blockdiagramm eines nicht beschränkenden beispielhaften Flashspeichersystems **100** gemäß manchen Ausführungsbeispielen vorliegender Erfindung. Das beispielhafte System **100** umfasst ein Speichermodul **120** für die Speicherung von Daten und einen Hostrechner **110** (Beispiele des Hostrechners **110**: ein Mikrocomputer, ein Smartcard-Terminal, eine digitale Kamera, ein Mobiltelefon, ein PDA

oder jedes andere Gerät), der mit dem Speichermodul **120** über eine Hostschnittstelle **180** kommuniziert.

[0056] Das Speichermodul **120** umfasst einen Flashspeicher **130** beliebigen Typs sowie eine Steuerung **140**, die auf den Flashspeicher **130** gemäß den über die Hostschnittstelle **180** empfangenen Lese- und/oder Schreibbefehlen zugreift. Für das in Abb. 1 gezeigte Beispiel umfasst die Steuerung **140** ein CPU **150**, eine ROM **160** (in der der von der CPU **150** ausgeführte Code hinterlegt ist) und ein RAM **170**, das von der CPU **150** verwendet wird, um die Ausführung des Code durch die Steuerung **140** zu unterstützen.

[0057] Dieses Blockdiagramm des nicht beschränkenden Beispiels gemäß Abb. 1 ist repräsentativ für typische nicht volatile Speichermodule wie SecureDigital Flashspeicherkarten oder mobile USB Flashlaufwerke.

[0058] Abb. 5 ist ein Blockdiagramm eines weiteren nicht beschränkenden Flashspeichersystems **220** gemäß manchen Ausführungsbeispielen vorliegender Erfindung. Das exemplarische Flashspeichersystem **220** umfasst eine Standard-CPU **250**, ein RAM **260**, den Flashspeicher **280**, eine Busschnittstelle **290** zum Flashspeicher **280**, einen Boot-ROM **270**, eine Speichervorrichtung **300** und einen Bus **240**, die die verschiedenen anderen Komponenten miteinander verbinden. Wenn das System **220** startet, lädt das System aus dem ROM **270**; dann werden der Computercode und die Daten aus dem Speichermedium **300** in die RAM **260** geladen. Auch Emulationscode für die Steuerung des Flashspeichers **280** wird aus dem Speichermedium **300** geladen. Die Bus-Schnittstelle **290** greift auf den Flashspeicher **280** gemäß den vom CPU **250** empfangenen Lese- und/oder Schreibbefehlen zu. Das Speichermedium **300** ist ein Beispiel eines computerlesbaren Speichermediums, das Computercode für die Umsetzung der Verfahren gemäß vorliegender Erfindung trägt. Typischerweise ist das Speichermedium **300** eine Festplatte oder eine Flashspeichervorrichtung. Weitere Beispiele solcher computerlesbarer Speichermedien sind unter anderen CDs, DVD, Disketten etc. Im Unterschied zu dem Ausführungsbeispiel des Flashspeichersystems gemäß Abb. 1 hat dieses beispielhafte Flashspeichersystem **220** keine Flashspeichersteuerung (die das Flashspeichersystem kontrolliert). Stattdessen lädt die CPU **250** den Steuerungsemulationscode aus dem Massenspeicher **300** auf die RAM **260**, und dann führt die CPU **250** den Code aus RAM **260** aus, um die Steuerung **140** gemäß Abb. 1 zu emulieren. Die Speicherungen der Flashmanagementtabellen und deren Wiederherstellung und Rekonstruktion nach dem Einschalten sowie weitere Flashmanagementfunktionen werden alle durch den von der CPU **250** ausgeführten Emulationscode umgesetzt.

[0059] Abb. 6 ist ein Blockdiagramm eines weiteren nicht beschränkenden Flashspeichersystems **320** gemäß manchen Ausführungsbeispielen vorliegender Erfindung. Das exemplarische System **320** umfasst eine Standard-CPU **350**, ein RAM **360**, ein Flashspeichermodul **330**, eine Flashspeichersteuerung **310**, einen Flashspeicher **380**, eine Bus-Schnittstelle **390** zum Flashspeichermodul **330**, eine Boot-ROM **370**, ein Speichermedium **400** und einen Bus **340**, die die verschiedenen anderen Komponenten miteinander verbinden. Wenn das System **329** startet, lädt das System aus ROM **379**; dann werden der Computercode und die Daten aus dem Speichermedium **400** auf die RAM **360** geladen. Ebenfalls aus dem Speichermedium **400** geladen wird der Emulationscode, der das Flashspeichermodul **330** steuert. Die Bus-Schnittstelle **390** greift auf den Flashspeicher **380** gemäß den von der CPU **350** empfangenen Lese- und/oder Schreibbefehlen zu. Wie das Speichermedium **300** ist auch das Speichermedium **400** ein Beispiel eines computerlesbaren Speichermediums, das Computercode für die Ausführung der Verfahren gemäß vorliegender Erfindung vorhält. Im Unterschied zum Flashspeichersystem gemäß Abb. 5 hat dieses beispielhafte Flashspeichersystem **320** auch in seinem Flashspeichermodul **330** eine Flashspeichersteuerung **310**, die mit CPU **350** zusammenwirkt, um das Flashspeichersystem **380** zu steuern. Die CPU **350** lädt Steuerungsemulationscode aus dem Speichermedium **400** auf RAM **360**, und dann führt die CPU **350** den Code aus RAM **360** aus, um einige der Funktionen der Steuerung **140** gemäß Abb. 1 zu emulieren. Die Speicherung der Flashmanagementtabellen sowie deren Wiederherstellung und Rekonstruktion nach dem Einschalten und weitere Flashmanagementfunktionen werden gemeinsam von der Steuerung **310** und CPU **350** ausgeführt.

Flashmanagementtabellen

[0060] Zu beachten ist, dass das Flashspeichersystem **100** typischerweise eine oder mehrere Flashmanagementtabellen in einem volatilen Speicher hinterlegt, zum Beispiel in RAM **170** des Speichermediums **120**, in der RAM des Hostgeräts **110** oder an einem anderen geeigneten Ort. Ein nicht beschränkendes Ausführungsbeispiel einer solchen Flashmanagementtabelle ist eine Translationstabelle, die eine Adressentranslation aus einer virtuellen Blocknummer in eine physische Blocknummer vornimmt. Dies ist ein Mapping, das in vielen Flashmanagementsystemen vorhanden ist, wie zum Beispiel im System gemäß US-Patent 5,937,425. Es ist zu beachten, dass die gleichen Konzepte und Verfahren auch für viele andere Arten von Flashmanagementtabellen anwendbar sind, so zum Beispiel auf eine Tabelle zugeordneter Blöcke, die darstellt, welche Blöcke zurzeit zugeteilt und nicht frei für die Verwendung verfügbar

sind und welche Blocke nicht zugeteilt sind, sowie auf eine Tabelle, die ein Mapping aus einer virtuellen Blocknummer in eine Gruppe einer oder mehrerer physischer Blocknummern abbildet.

[0061] Die im Flashspeicher gespeicherten Daten können sich im Laufe der Zeit ändern; und auch verschiedene zugehörige Daten, die sich auf den Flashspeicher beziehen, können sich ebenfalls ändern. Der „Status" des Flashspeichersystems ändert sich im Zeitverlauf, wenn verschiedene Ereignisse (zum Beispiel Schreibbefehle, Verwaltungsoperationen etc.) des Flashmanagementsystems eintreten.

[0062] Zu beachten ist, dass jede Flashmanagementtabelle einen oder mehrere Aspekte des gesamten „Status" des Flashmanagementsystems abbildet. Folglich stellt eine gegebene Flashmanagementtabelle oder ein Satz solcher Tabellen nicht unbedingt den gesamten Status des Systems dar, sondern nur einen oder mehrere Aspekte des Systems. Im Fall der vorstehend erwähnten Tabelle freier Blöcke, ist das Wissen, welche Blöcke frei sind und welche nicht, in jedem Fall nicht ausreichend, um den gesamten Status des Systems zu definieren. Ein nicht freier Block kann eine einzige verwendete Seite umfassen, oder manche bzw. alle Seiten eines nicht freien Blocks können mit gültigen Daten gefüllt sein. Das wird von der Tabelle freier Blöcke nicht abgebildet, sondern entweder durch andere Flashmanagementtabellen oder sonstige Mittel. Dennoch bildet die genannte Tabelle einen Aspekt des Systemstatus ab und erfüllt daher die Definition einer Flashmanagementtabelle.

[0063] Werden bestimmte Flashspeichertabellen, die einen „aktuellen Status" oder einen „früheren Status" oder einen „späteren Status" oder den „jüngsten Status" des Flashspeichersystems abbilden, in einem volatilen oder nicht volatilen Speicher hinterlegt, so wird damit folglich ein „früherer" oder „aktueller" oder „späterer" oder „jüngster" Status der Aspekte des Flashspeichersystems abgebildet, der in der jeweiligen Flashspeichertabelle abgebildet ist.

[0064] Im Zeitverlauf bewegt sich eine Flashmanagementtabelle durch eine Folge von Zuständen, von denen jeder zu einem gegebenen Zeitpunkt einen oder mehrere Aspekte des Flashmanagementsystems zu diesem Zeitpunkt abbildet. Der Aspekt des Systemstatus, der von der Tabelle modelliert ist, wechselt zwischen diskreten Zuständen mit eindeutigen Übergangspunkten, die den Ereignissen des Flashspeichersystems entsprechen.

Beispiel einer Flashmanagementtabelle

[0065] Abb. 2A zeigt den Inhalt einer exemplarischen Flashmanagementtabelle, die zu einem ersten Zeitpunkt ($t = t_1$) virtuelle Blocknummern auf physische Blocknummern mappiert. Die Tabelle ist nach virtuellen Blocknummern indexiert und führt die physische Blocknummer auf, die zu diesem Zeitpunkt die Daten des entsprechenden virtuellen Blocks enthält. In praktischen Implementierungen besteht oft keine Notwendigkeit, Raum für die Speicherung der virtuellen Blocknummern zuzuteilen, da die Tabelle durch diese virtuellen Blocknummern geordnet ist und sich daher direkt auf den richtigen Eintrag indexieren lässt. Doch zwecks besserer Übersicht ist die Tabelle so dargestellt, als seien beide Spalten physisch abgebildet.

[0066] Der in Abb. 2A dargestellte Status des Mapping virtuell-auf-physisch ist so, dass wenn auf den virtuellen Block Nummer 2 zugegriffen werden muss, die Tabelle uns mitteilt, dass die entsprechende Nummer des physischen Blocks 172 lautet. Zu einem bestimmten Zeitpunkt können an das Flashmanagementsystem übermittelte Befehle (oder interne Verwaltungsfunktionen des Systems, die ohne externe Befehle erfolgen, wie das Einsammeln redundanter Daten) Änderungen des Mapping auslösen. Zum Beispiel kann ein Befehl, der neue Daten in den virtuellen Block Nummer 2 schreibt und daher die vorherigen Daten des Blocks Nummer 2 überschreibt, das Flashmanagementsystem veranlassen, dem virtuellen Block Nummer 2 einen anderen physischen Block zuzuordnen, in dem die neuen Daten dann gespeichert werden.

[0067] Abb. 2B zeigt die Inhalte der Flashmanagementtabelle virtuell-auf-physisch zu einem späteren Zeitpunkt ($t = t_2$) nach Eintritt dieser Änderung. Jetzt entspricht der virtuelle Block Nummer 2 in der Tabelle dem physischen Block Nummer 777 und nicht mehr dem physischen Block 172. Die Tabelle ist also von einem ersten in einen zweiten Status gewechselt. Jede Änderung der Inhalte einer Flashmanagementtabelle ist definiert als Änderung des Status der Tabelle.

[0068] Zu beachten ist, dass nicht jede Schreiboperation im Flashmanagementsystem eine Statusänderung in allen Flashmanagementtabellen des Flashmanagementsystems nach sich zieht. Sind zum Beispiel im physischen Block Nummer 172 einige Seiten unbelegt und ergeht ein Schreibbefehl an den virtuellen Block Nummer 2, so dass gemäß den Algorithmen des Flashmanagementsystems die neuen Daten in diesen unbelegten Seiten des physischen Blocks 172 gespeichert werden, erfolgt keine Änderung des Mapping virtuell-auf-physisch, und die Tabelle bleibt nach dieser Schreiboperation unverändert. Folglich ist die Rate der Änderungen des Status einer Flashmanagementtabelle generell langsamer als die Rate der im System ausgeführten Vorgänge. Hinzu kommt: werden im gleichen Flashmanagementsystem mehrere Flashmanagementtabellen vorgehalten (von denen jede einen anderen Aspekt des

Systemstatus abbildet), kann jede dieser Tabellen ihren Status zu unterschiedlichen Zeitpunkten ändern.

Speichern künftiger Informationsdatenstrukturen, die beim Wakeup nützlich sind

[0069] Wie bereits erklärt, ist es bei der Initialisierung des Systems nach dem Einschalten oft erforderlich, dass die Steuerungssoftware im RAM (z. B. RAM **170**) eine vollständig aktualisierte Kopie jeder Flashmanagementtabelle erstellt, die die Steuerungssoftware verwendet. Um ein schnelles Hochfahren des Systems zu gewährleisten, ist es generell nicht angemessen, die Inhalte der Tabellen lediglich aus über die Blöcke des Speichersystems verstreuten Daten wiederherzustellen, da das zu langsam ist. Auch die umgekehrte Option, jedes Mal, wenn sich der Tabellenstatus ändert, eine Kopie der Tabelle im Flashspeicher abzuspeichern, ist nicht angemessen, da das Speichern einer Kopie einer Tabelle im Flashspeicher jedes Mal, wenn der Tabellenstatus sich ändert, viel Kapazität beansprucht und die Systemleistung mindert.

[0070] Die Techniken der verschiedenen Ausführungsbeispiele vorliegender Erfindung bieten einen Kompromiss zwischen diesen beiden Vorgehensweisen des Standes der Technik. So wird eine Flashmanagementtabelle generell nur bei manchen Statusänderungen im Flashspeicher hinterlegt, und nicht jedes Mal, wenn eine Statusänderung eintritt.

[0071] Das zeigt **Abb.** 2, ein Flussdiagramm einer beispielhaften Routine für die Pflege von Datenstrukturen des Flashmanagementsystems im Flashspeicher gemäß einigen Ausführungsbeispielen vorliegender Erfindung. Für die vorliegenden Zwecke umfasst der Ausdruck „Pflege im Flashspeicher" auch das Speichern im Flashspeicher. Die Pflege im Flashspeicher umfasst generell die Pflege der Tabelle und/oder der relevanten Daten für die Befüllung der Tabelle im volatilen Speicher/RAM und, zu verschiedenen Zeitpunkten, das Speichern einer aktualisierten Version der Tabelle (d. h. einer Tabelle, die auf einen aktuellen Status des Flashspeichersystem synchronisiert ist) im Flashspeicher.

[0072] Gemäß dem nicht beschränkenden Ausführungsbeispiel der **Abb. 3** ist zu beachten, dass in nicht beschränkenden Ausführungsbeispielen das System zu manchen Zeitpunkten inaktiv ist und **206** auf ein nächstes Ereignis wartet. Nach Bearbeitung **210** eines Ereignisses des Flashmanagementsystems, werden generell je nach eingetretenem Ereignis eine oder mehrere Tabellen im volatilen Speicher aktualisiert **214**. Diese Tabelle(n) können nach einem gegebenen Ereignis im Flashspeicher aktualisiert werden oder nicht.

Beschreibung einer beispielhaften künftigen Informationsdatenstruktur

[0073] In Situationen, in denen die Flashmanagementtabelle nach Eintritt einiger, aber nicht aller Ereignisse gespeichert wird, ist es nützlich, auch im Flashspeicher eine Datenstruktur zu pflegen, die Informationen enthält, die die Handhabung künftiger Ereignisse in Bezug auf die Tabelle beeinflussen. In Ausführungsbeispielen enthält diese künftige Informationsdatenstruktur eine Vielzahl von Datensätzen, wobei jeder Datensatz Daten umfasst, die ein künftiges Ereignis im Flashmanagementsystem beeinflussen. Obgleich dies keine Beschränkung vorliegender Erfindung begründet, ist anzumerken, dass diese künftigen Informationen generell nicht nach jedem Ereignis im Flashspeicher aktualisiert werden. Die künftige Informationsdatenstruktur ist nützlich für den Erhalt der Datenintegrität in Situationen, in denen (eine) aktualisierte Tabelle(n) (d. h. aktualisiert gemäß dem letzten Ereignis im Flashspeichersystem) nicht immer im Flashspeicher hinterlegt wird/werden.

[0074] Für Ausführungsbeispiele, in denen Datenintegrität gesichert wird (d. h. selbst im Falle eines Stromausfalls), ist generell zu jedem beliebigen Zeitpunkt einer der folgenden Zustände gegeben: a) eine gemäß einem jüngsten Ereignis aktualisierte Tabelle ist im Flashspeicher hinterlegt, b) eine gemäß einem früheren Zustand aktualisierte Tabelle (d. h. früher als ein jüngstes Ereignis, so dass andere Ereignisse seit dem „früheren" Ereignis im Flashspeichersystem eingetreten sind) wird im Flashspeicher hinterlegt, und die im Flashspeicher gespeicherte künftige Informationsdatenstruktur enthält Informationen bezüglich aller Ereignisse, die seit dem „früheren Ereignis" eingetreten sind.

[0075] Ein Ausführungsbeispiel der künftigen Informationsdatenstruktur ist eine geordnete Liste, die Informationen für die Handhabung der nächsten paar Ereignisse enthält, die den Status des Systems beeinflussen (d. h. den Status des „mindestens einen Aspekts des Systems", der in der einen oder mehreren Flashmanagementtabelle(n) gespeichert ist), sei es ein externer Schreibbefehl, ein interner Verwaltungsvorgang oder ein sonstiges Ereignis, das den Status des Systems beeinflusst.

[0076] Zu beachten ist, dass eine künftige Informationsdatenstruktur an jedem Ort im Flashspeicher hinterlegt werden kann, und nicht unbedingt in physischer Nachbarschaft zu der Flashmanagementtabelle, auf die sich die künftige Informationsdatenstruktur bezieht.

[0077] Wie in **Abb. 3** dargestellt, wird/werden **230** die Flashmanagementtabelle(n) für manche, aber nicht alle Ereignisse im Flashmanagementsystem (d. h. nur Ereignisse, für die eine Bedingung für „Tabelle

speichern" erfüllt ist **226** – diese Bedingung wird nachstehend erörtert) im Flashspeicher gespeichert. In manchen Ausführungsbeispielen wird jedes Mal, wenn (a) eine/mehrere Flashmanagementtabelle(n) im Flashspeicher hinterlegt werden, auch eine aktualisierte künftige Informationsdatenstruktur gespeichert **234**. Zu diesem Zeitpunkt gilt/gelten die im Flashspeicher gespeicherten Flashmanagementtabelle(n) als „auf den neusten Stand" oder „das jüngste Ereignis" synchronisiert. Obgleich nicht ausdrücklich in Abb. 3 dargestellt, ist zu beachten, dass es ratsam ist (jedoch für den Einsatz vorliegender Erfindung nicht wesentlich), eine Kopie der zuletzt gespeicherten künftigen Informationsdatenstruktur im RAM-Speicher zu hinterlegen, damit die Bearbeitung künftiger Ereignisse erfolgen kann, ohne dass Zugang zum nicht volatilen Speicher nötig ist, um die künftige Informationsdatenstruktur abzurufen.

[0078] In Bezug auf Schritt **230** ist anzumerken: wird eine den aktuellen Status des Flashspeichersystems abbildende Tabelle im Flashspeicher gespeichert, ist dies vorliegend definiert als „Synchronisieren der Tabelle im Flashspeicher auf einen aktuellen Status".

[0079] Insgesamt wird der Prozess, in dem die Flashmanagementtabelle(n) nach diversen Ereignissen (aber nicht notwendigerweise allen Ereignissen) im Flashspeicher gespeichert wird/werden vorliegend als „Pflege der Tabelle im Flashspeicher" definiert. Zu verschiedenen Zeitpunkten wird somit eine unterschiedliche Version der Flashmanagementtabelle(n) im Flashspeicher hinterlegt (wobei generell jede folgende Version einen späteren Zustand des Flashspeichersystems abbildet). Es ist für die „Pflege der Tabelle im Flashspeicher" daher nicht erforderlich, für einen gegebenen Zeitpunkt die zuletzt im Flashspeicher hinterlegte Tabelle aus den aktuellen Status des Flashspeichersystems zu synchronisieren. Abb. 3 zeigt, dass es generell Zeiträume gibt, in denen die zuletzt gespeicherte(n) Flashspeichertabelle(n) einen früheren Zustand des Flashspeichersystems abbilden (d. h. einen Status des Systems vor Eintritt jüngerer Ereignisse im Flashspeichersystem).

[0080] Es ist zu beachten, dass die künftige Informationsdatenstruktur später aus dem Flashspeicher abgerufen werden kann, zum Beispiel nach dem Einschalten des Flashsystems. Durch Prüfung der Datenstruktur (wie nachstehend beschrieben) wird zu jedem gegebenen Zeitpunkt festgestellt, ob die hinterlegte(n) Flashmanagementtabelle(n) den jüngsten Status des Systems (d. h. den in der Tabelle abgebildeten Aspekt des Status) abbildet/n.

[0081] Wir wenden uns nun der beispielhaften Flashmanagementtabelle gemäß Abb. 2A und Abb. 2B zu, um das nicht beschränkende Ausführungsbeispiel einer künftigen Informationsdatenstruktur zu beschreiben. Die Flashmanagementtabelle gemäß Abb. 2A und Abb. 2B bietet ein Mapping von virtuellen Blocknummern zu korrespondierenden physischen Blocknummern. Während des Systembetriebs ändert sich der Status der Tabelle jedes Mal, wenn die Flashmanagementsoftware einen freien physischen Block einem gegebenen virtuellen Block zuordnet, wobei gleichzeitig der zuvor diesem virtuellen Block zugeordnete physische Block frei gestellt wird.

[0082] Die der Tabelle gemäß einem Ausführungsbeispiel vorliegender Erfindung zugeordnete künftige Informationsdatenstruktur ist eine geordnete Liste der aktuell freien Blöcke. Die Liste definiert die exakte Reihenfolge, in der die freien Blöcke künftig zugeteilt werden. Anders gesagt: das nächste Mal, wenn die Flashmanagementsoftware einen freien Block zuteilt, ist garantiert, dass die Flashmanagementsoftware den Block an erster Stelle in der Liste auswählen wird. Das zweite Mal, wenn die Flashmanagementsoftware einen freien Block zuteilt, ist garantiert, dass die Flashmanagementsoftware den Block an zweiter Stelle in der Liste zuteilen wird, usw.

[0083] Es ist zu beachten, dass jeder physische Block generell in sich die Nummer des virtuellen Blocks enthält, dem der betreffende physische Block aktuell zugeordnet ist. Dies ermöglicht der Flashmanagementsoftware, den aktuellen Status der Tabelle aus der hinterlegten (und nicht aktuellen) Version der Tabelle und ihrer diesbezüglichen Liste künftiger Informationen zu rekonstruieren, wie nachstehend in der Beschreibung des Wakeup-Prozesses des Systems erörtert wird.

Regeln für die Bestimmung des Zeitpunkts der Speicherung (einer) aktualisierter/n Flashmanagementtabelle(n) im Flashspeicher

[0084] Jede Regel für die Bestimmung eines Ereignisses, für das die aktualisierte Tabelle im Flashspeicher gespeichert wird – d. h. die „Speicherbedingung" gemäß Schritt **226** –, ist im Schutzumfang vorliegender Erfindung inbegriffen.

[0085] Der Schutzumfang vorliegender Erfindung umfasst eine Reihe von Regeln für die Bestimmung, wann (eine) aktualisierte Flashmanagementtabelle(n) gemäß bestimmten Ausführungsbeispielen vorliegender Erfindung im Flashspeicher zu speichern ist.

[0086] In einem ersten Ausführungsbeispiel wird eine Tabelle bei jeder Nten Änderung des Status der Tabelle gespeichert, wobei N vorgegeben ist. N kann so klein wie 2 sein, wenn die Datensicherungsbelastung niedrig ist, oder so groß wie 100 oder sogar mehr, zum Beispiel, wenn die Datensicherungsbelastung hoch ist. Daher wird eine Zählervariable auf Null gesetzt. Nach jedem Ereignis wird die Zählervariable

erhöht. Überschreitet die Zählervariable den vorgegebenen Wert N, werden eine oder mehrere Tabellen im Flashspeicher gespeichert, und die diesen Tabellen zugeordnete künftige Informationsdatenstruktur wird neu berechnet und zusammen mit der/den Tabelle(n) im Flashspeicher hinterlegt.

[0087] In einem zweiten Ausführungsbeispiel wird die Bedingung für „Tabelle speichern" gemäß der durch ein jüngstes Ereignis ausgelösten Änderung des Status bestimmt. Für diesen Zweck werden Statusänderungen als „geringfügige" oder „bedeutende" Änderungen klassifiziert. Änderungen im Tabellenstatus, die geringfügig sind, lösen kein sofortiges (d. h. vor Bearbeitung **210** des nächsten Ereignisses) Speichern der Tabelle aus (d. h. den Zweig „NO" nach Schritt **226**), während bedeutende Änderungen sofortiges Speichern auslösen. Ein nicht beschränkendes Beispiel einer Klassifizierung von Änderungen in geringfügig und bedeutend im Falle einer Flashmanagementtabelle eines freien Blocks ist, dass eine Änderung, die einen nicht freien Block in einen freien Block verwandelt, als bedeutend gilt.

[0088] In einem dritten Ausführungsbeispiel wird/werden die Tabelle(n) periodisch gespeichert, sobald ein vorgegebenes Zeitintervall abgelaufen ist. Generell besteht ein Kompromiss zwischen der Häufigkeit der Tabellenaktualisierung und dem Umfang der verfügbaren Systemkapazität, der durch das Speichern der Flashmanagementtabelle(n) im Flashspeicher beansprucht wird. Andererseits bedeutet das Speichern der Managementtabelle(n) öfter, dass im Durchschnitt eine beim Wakeup aus dem Flashspeicher abgerufene Tabelle voraussichtlich aktualisierter sein wird, so dass der Wakeup-Vorgang schneller erfolgt. Zu beachten ist, dass jedes Zeitintervall im Schutzumfang der vorliegenden Erfindung inbegriffen ist. In nicht beschränkenden Ausführungsbeispielen beträgt das Zeitintervall zwischen einer Zehntelsekunde und fünf oder mehr Minuten.

[0089] In einem vierten Ausführungsbeispiel wird die Häufigkeit der Tabellenaktualisierung gemäß der Verfügbarkeit von Systemkapazität bestimmt. In einem Beispiel, in dem die Steuerung **140** viele Lese-/Schreib-/Löschbefehle bearbeitet, oder in einer Zeit, in der viele Verwaltungsabläufe erfolgen, wird/werden die Flashmanagementtabelle(n) weniger häufig im Flashspeicher gesichert, um Systemkapazität zu sparen. In Zeiten „geringer Beanspruchung", wenn das System ansonsten ruht oder geringer Beanspruchung unterliegt, ist es generell möglich, die Flashmanagementtabelle(n) häufiger ohne bedeutende Auswirkungen auf die Leistung des Flashsystems im Flashspeicher zu speichern.

[0090] Weitere Ausführungsbeispiele der Regeln zur Bestimmung der Zeitpunkte, an denen eine Speicherung einer Flashmanagementtabelle im Flashspeicher erfolgt, sind ebenfalls möglich.

[0091] Wird/werden (eine) Flashmanagementtabelle(n) im Flashspeicher gesichert, sollte die zugehörige künftige Informationsdatenstruktur vorzugsweise ausreichend Information enthalten, um die Bearbeitung aller künftigen Ereignisse zu gewährleisten, bis die nächste Speicherung erfolgt. Es ist ratsam, eine Sicherheitsmarge vorzusehen und für mehr Ereignisse Vorsorge zu treffen als bis zum nächsten Speichervorgang erwartet werden. Im Kontext des aktuellen Ausführungsbeispiels können wir eine Liste der freien Blöcke erstellen, die als nächstes zu verwenden sind, die mehr freie Blöcke enthält, als bis zum nächstes Speichervorgang als erforderlich erachtet. Doch die Erfindung hängt nicht davon ab, dass so verfahren wird, und bearbeitet alle Ereignisse korrekt, selbst wenn sich schließlich herausstellt, dass alle gespeicherten künftigen Informationen bereits verbraucht worden sind und ein zusätzliches Ereignis empfangen wird. In diesem Fall führen wir einfach, als sei eine „Speicherbedingung" erfüllt, eine umgehende Synchronisierung der Tabelle auf den Flash durch. Aus Gründen der Übersichtlichkeit ist dieser Fall des Verbrauchs künftiger Informationen in **Abb. 3** nicht dargestellt.

[0092] Da nicht jeder Status eine Speicherung der Tabelle auslöst, entsteht schließlich eine Lücke oder Fehlanpassung zwischen der letzten gespeicherten Kopie einer Tabelle und der zuletzt aktualisierten Kopie der Tabelle im RAM. Bei jedem Speichern der Tabelle wird die Lücke eliminiert, und die beiden Kopien (d. h. die Kopie im volatilen RAM-Speicher und die im Flashspeicher gesicherte Kopie) werden identisch, doch darauf folgende neue Statusänderungen, die keine Speicherung der Tabelle auslösen, erzeugen wieder eine Lücke.

Wakeup

[0093] In manchen Ausführungsbeispielen wird/werden, wenn eine Anwendung ordnungsgemäß beendet wird, aktualisierte Flashmanagementtabelle(n) (d. h. (a) Flashmanagementtabelle(n), die gemäß den jüngsten Ereignissen im Flashspeichersystem aktualisiert wurde(n)) im Flashspeicher gespeichert. Fährt das Flashspeichersystem hoch, wird/werden diese Flashmanagementtabelle(n) aus dem Flashspeicher abgerufen.

[0094] Erfolgt kein ordnungsgemäßes Beenden, ist es beim Hochfahren möglich, die „veraltete(n)" Flashmanagementtabelle(n) aus dem Flashspeicher in den volatilen Speicher zu laden und dann gemäß in der künftigen Informationsdatenstruktur gespeicherten Informationen die veraltete Flashmanagementtabelle im volatilen Speicher zu aktualisieren.

[0095] Ausführungsbeispiele der vorliegenden Er-

findung machen daher die Invalidierung veralteter Tabellen wie in Lasser US 6,510,488 vorgesehen überflüssig.

[0096] Abb. 4 ist ein Flussdiagramm einer beispielhaften Wakeup-Routine gemäß einigen Ausführungsbeispielen vorliegender Erfindung. Abb. 4 zeigt eine beispielhafte Wakeup-Routine im Kontext des vorstehenden Beispiels der Tabelle in Abb. 2A und Abb. 2B, wobei die künftige Informationsdatenstruktur eine geordnete Liste freier Blöcke ist, die als nächste zugeteilt werden. Nach Einschalten von **410** ruft das Flashmanagementsystem **414** die gespeicherte Kopie einer Flashmanagementtabelle auf. Dann ruft das Flashmanagementsystem **418** die zugehörige Liste künftiger Informationen auf, die zum gleichen Zeitpunkt gespeichert wurde, als die Flashmanagementtabelle im Flashspeicher hinterlegt wurde.

[0097] Als nächstes ruft **422** das System die erste physische Blocknummer in der Liste auf und setzt einen Marker **426** auf den ersten Eintrag in der Liste.

[0098] Wurde das System sofort nach der letzten Speicherung der Flashmanagementtabelle und vor Eintritt einer weiteren Änderung des Status der Tabelle abgeschaltet (zum Beispiel nach „ordnungsgemäßem Herunterfahren" oder in jeder anderen Situation), sollte der erste Block in der Liste noch frei sein. Sind jedoch nach der letzten Speicherung der Tabelle und vor dem Abschalten des Systems eine oder mehrere Statusänderungen der Flashmanagementtabelle(n) eingetreten, sollte der erste Block in der Liste jetzt belegt sein.

[0099] Das Einschalten der Flashmanagementsoftware kann feststellen, ob ein physischer Block aktuell belegt ist oder nicht. Dies lässt sich unabhängig von jeder Flashmanagementtabelle durch Prüfung der Inhalte eines oder mehrerer Kontrollfelder innerhalb des Blocks bestimmen. In manchen Flashmanagementsystemen reicht es aus, das Kontrollfeld „korrespondierende virtuelle Blocknummer" zu suchen. Ist dort eine gültige virtuelle Blocknummer eingetragen, ist der Block belegt; enthält das Feld keine gültige virtuelle Blocknummer, ist der Block nicht belegt. In anderen Flashmanagementsystemen erfordert die Bestimmung, ob ein physischer Block zurzeit frei ist oder nicht, die Prüfung von mehr als einem Kontrollfeld, doch die Bestimmung ist in jedem Fall relativ leicht und rasch umsetzbar.

[0100] Durch Anwendung der beschriebenen Techniken zur Bestimmung des „freien" Status des ersten physischen Blocks in der Liste künftiger Informationen kann die Flashmanagementsoftware daher bestimmen, ob eine Diskrepanz zwischen der empfangenen Tabelle und der „wahren" Tabelle besteht, die den aktuellen Status des Systems abbilden würde.

[0101] Stellt sich heraus, dass der erste Block in der Liste belegt ist, wissen wir, dass die aufgerufene Tabelle aktualisiert werden sollte. Wir rufen dann **434**, die aktuelle korrespondierende virtuelle Blocknummer des Blocks, aus dem ersten Block auf. Dies gestattet uns, die Tabelle in RAM **438** zu aktualisieren, um die aktuelle Korrespondenz zwischen den virtuellen und physischen Blocknummern abzubilden. Der in der aufgerufenen Tabelle als dem virtuellen Block (von dem jetzt bekannt ist, dass er durch den ersten Block in der Liste ersetzt worden ist) korrespondierend angezeigte physische Block wird von der Tabelle nicht mehr angezeigt, da dieser physische Block aktuell nicht belegt ist.

[0102] Als nächstes wird der Zeiger in der Liste weiter geschoben **442**, und der gleiche Ablauf wird für den nächsten physischen Block in der Liste wiederholt. So lange der gesichtete Block sich als belegt erweist, wird die Tabelle im RAM aktualisiert, um das Ereignis der Zuteilung des betreffenden Blocks abzubilden.

[0103] Schließlich erreichen wir einen Punkt, an dem der gesichtete physische Block sich als frei erweist. Dies impliziert, dass keine weiteren Aktualisierungen der Tabelle im RAM erforderlich sind und dass die Tabelle den Systemstatus jetzt zutreffend abbildet. Jetzt wird die aktualisierte Tabelle im Flashspeicher **446** gespeichert. Gleichzeitig wird auch eine neu berechnete Liste künftiger Informationen (die die nächsten zuzuteilenden freien Blöcke anzeigt) im Flashspeicher hinterlegt, so dass nach erneutem Abschalten des Systems die gleiche Wakeup-Routine wieder den korrekten Status der Tabelle wiederherstellt. Zu beachten ist, dass Schritt **446** des Speicherns der Tabelle im Flashspeicher nicht wirklich erforderlich ist, wenn festgestellt wird, dass kein Block aus der Liste künftiger Informationen zugeteilt worden ist, da in diesem Fall die gespeicherte Tabelle bereits aktualisiert ist. Aus Gründen der besseren Übersicht ist dies in Abb. 4 nicht dargestellt.

[0104] Nachdem festgestellt ist, dass die Flashspeichertabelle(n) im volatilen und im nicht volatilen Speicher aktualisiert ist/sind, kann **450** mit dem regulären Betrieb des Flashspeichersystems fortgefahren werden.

[0105] Interessant ist die Beobachtung, dass die Verfahren gemäß vorliegender Erfindung in gewisser Weise das genaue Gegenteil der Verfahren von Lasser '056 sind. Beide Verfahren haben viele Gemeinsamkeiten – sie aktualisieren beide die Flashmanagementtabellen im Flashspeicher nur gelegentlich und gestatten dadurch die Entstehung einer Lücke zwischen dem gespeicherten Status und dem aktuellen Status. Beide behandeln diese Lücke mittels „Playback" eines iterativen Aktualisierungsprozesses zum Zeitpunkt des Einschaltens auf Grundlage der

im Flash gespeicherten Informationen, die eindeutig die Ereignisse definieren, die zur Entstehung der Lücke geführt haben. Doch besteht ein grundlegender Unterschied zwischen den Verfahren bezüglich dieser Informationen. Lasser '056 bearbeitet die eingehenden Ereignisse auf Grundlage der spezifischen Algorithmen des Flashmanagementsystems, welche auch immer diese sein mögen, und hinterlegt dann diese Ereignisse im Flash in einer Weise, die abbildet, wie diese Ereignisse bearbeitet worden sind. Die vorliegende Erfindung hinterlegt erst im Flash, wie eingehende Ereignisse künftig bearbeitet werden sollen (wieder auf Grundlage der Algorithmen des Flashmanagementsystems, aber vor Eintritt der Ereignisse), und bearbeitet die Ereignisse dann, wenn sie tatsächlich eintreten, gemäß den hinterlegten Bearbeitungsentscheidungen.

Allgemeine Erörterung zur Leistung

[0106] Ein Vorteil bestimmter Ausführungsbeispiele vorliegender Erfindung besteht darin, dass das Lesen der gespeicherten Kopie einer Flashmanagementtabelle und die Aktualisierung der Tabelle in Bezug auf die Ereignisse, die noch nicht in der gespeicherten Tabelle abgebildet sind, sehr viel weniger Zeit benötigt als die Neuerstellung der Tabelle durch Scannen der vielen Blöcke des Speichersystems. Wurde das System nach Speicherung der Tabelle vor Eintritt weiterer Statusänderungen abgeschaltet (zum Beispiel im Falle ordnungsgemäßen Herunterfahrens oder wenn wir das „Glück" hatten, anzuschalten bevor neue Ereignisse eingetreten waren), erfolgt das Wakeup in manchen Ausführungsbeispielen schnell, so wie bei Lasser '488 im Fall des ordnungsgemäßen Herunterfahrens. Doch selbst wenn nach dem letzten Speichern der Tabelle einige Statusänderungen eingetreten sind, ist die für das Wakeup benötigte Zeit unter Umständen nicht so lang wie beim Verfahren gemäß Lasser '488, wenn kein ordnungsgemäßes Herunterfahren stattgefunden hat. In vielen Situationen müssen nur wenige Ereignisse entdeckt und ihre Folgen für den Status der Tabellen wiederhergestellt werden. Die genaue Zeit, die dafür generell benötigt wird, hängt von der Anzahl der Einträge ab, die mit der künftigen Informationsdatenstruktur bearbeitet werden müssen. Dies wiederum kann von der Häufigkeit abhängen, mit der die Tabelle im nicht volatilen Speicher hinterlegt wird. Je höher die Häufigkeit, desto weniger Einträge sind im Durchschnitt zu bearbeiten und desto schneller ist durchschnittlich der Wakeup-Vorgang. Andererseits: je höher die Häufigkeit des Speicherns, desto länger ist die für die Berechnung und Vorbereitung der künftigen Informationsdatenstruktur aufgewendete Zeit.

[0107] Die vorliegende Erfindung ähnelt in dieser Hinsicht in ihrer Leistung den Verfahren von Lasser '056. Doch dies gilt nur, wenn die für das Schreiben des Ereignisprotokolls in Lasser '056 aufgewendete Kapazitätsbelastung keine zusätzliche Kapazitätsbelastung bedingt, wie beispielsweise, wenn das Protokoll aus anderen Gründen ohnehin vorgehalten wird. Ist dies nicht der Fall, bieten die Verfahren gemäß vorliegender Erfindung eine bessere Gesamtleistung als Lasser '056, da das aufwändige Vorhalten des Protokolls nicht mehr erforderlich ist.

[0108] Während die vorstehenden Erörterungen sich im Wesentlichen auf eine einzelne Flashmanagementtabelle im Flashmanagementsystem beziehen, ist die Erfindung gleichermaßen auf viele Flashmanagementtabellen anwendbar, von denen jede einen unterschiedlichen Aspekt des Systemstatus abbildet. Sind viele Tabellen gegeben, wird jede Tabelle auf Grundlage der jeweils eigenen Speicherregeln der Tabelle gespeichert, und nicht unbedingt zu den gleichen Zeitpunkten. Bei Einschalten wird jede Flashmanagementtabelle mittels der beschriebenen Verfahren auf Grundlage der spezifischen Informationsdaten jeder Flashmanagementtabelle rekonstruiert. Es ist ebenfalls möglich, dass zwei oder mehr Tabellen eine gemeinsame künftige Informationsdatenstruktur besitzen, die zur Steuerung der Bearbeitung eingehender Ereignisse für diese multiplen Tabellen verwendet wird.

[0109] Es ist zu beachten, dass die vorliegende Erfindung sich nicht auf die vorstehend beschriebenen Ausführungsbeispiele beschränkt, die der Erklärung der Verfahren der Erfindung dienen. Die Erfindung ist gleichermaßen auch auf viele andere Typen von Flashmanagementtabellen und künftiger Informationsdatenstrukturen anwendbar, die alle im Schutzumfang der beanspruchten Erfindung liegen.

Zusätzliche Erörterung der vorliegend beschriebenen Systeme

[0110] Die vorliegend offen gelegten Techniken lassen sich mit jeder Kombination von Hardware, Firmware und Software umsetzen.

[0111] In einem nicht beschränkenden Ausführungsbeispiel werden das Speichern der Flashmanagementtabellen und deren Wiederherstellung und Rekonstruktion nach dem Einschalten alle durch die Steuerung **140** durchgeführt, oder genauer gesagt durch das Ausführen von in ROM **160** hinterlegtem Code durch CPU **150**. Dies ist jedoch nicht die einzige mögliche Systemarchitektur für den Einsatz vorliegender Erfindung. Es ist zum Beispiel auch möglich, die Verfahren der Erfindung durch im Hostrechner **110** ausgeführten Code umzusetzen, was der Fall ist, wenn das Speichermodul eine On-board-NAND Flashvorrichtung ist und keine selbständig operierende Steuerung vorhanden ist. Eine weitere Möglichkeit ist, dass die erfindungsgemäßen Verfahren wenigstens zum Teil durch den Hostrechner **110** und zum Teil durch die Steuerung **140** umgesetzt werden. Alle

diese Architekturen und viele weitere sind im Schutzumfang vorliegender Erfindung umfasst.

[0112] In der Beschreibung und den Ansprüchen vorliegender Anmeldung werden jedes der Verben „umfassen", „einschließen" und „haben" sowie deren Konjugate verwendet, um anzuzeigen, dass das oder die Objekte des Verbs nicht unbedingt eine vollständige Auflistung der Glieder, Komponenten, Elemente oder Teile des Subjekts oder der Subjekte des Verbs sind.

[0113] Alle vorliegend zitierten Veröffentlichungen sind durch Hinweis vollständig in vorliegende Offenlegung aufgenommen. Das Zitieren einer Veröffentlichung beinhaltet kein Eingeständnis, dass sie zum Stand der Technik zählt.

[0114] Die Artikel „der", „die", „das" dienen vorliegend dem Hinweis auf eines oder mehrere (d. h. mindestens eines) der grammatischen Objekte des Artikels. Zum Beispiel bezeichnet „ein Element" ein Element oder mehr als ein Element.

[0115] Der Ausdruck „einschließlich" bedeutet vorliegend „einschließlich, doch ohne Beschränkung hierauf" und wird mit dieser Formulierung austauschbar verwendet.

[0116] Der Ausdruck „oder" bedeutet vorliegend „und/oder" und wird damit austauschbar verwendet, es sei denn aus dem Zusammenhang geht klar eine andere Bedeutung hervor.

[0117] Der Ausdruck „wie beispielsweise" bedeutet und wird vorliegend austauschbar verwendet mit der Formulierung „wie beispielsweise, aber nicht beschränkt auf".

[0118] Die vorliegende Erfindung wurde vorstehend unter Verwendung detaillierter Beschreibungen von Ausführungsbeispielen derselben beschrieben, die als Beispiele vorgesehen sind und nicht dazu dienen, den Schutzumfang der Erfindung zu beschränken. Die beschriebenen Ausführungsbeispiele umfassen unterschiedliche Merkmale, von denen nicht jedes in allen Ausführungsbeispielen der Erfindung erforderlich ist. Manche Ausführungsbeispiele vorliegender Erfindung verwenden nur manche dieser Merkmale oder möglicher Merkmalskombinationen. Abwandlungen der Ausführungsbeispiele vorliegender Erfindung, die andere Kombinationen der in den beschriebenen Ausführungsbeispielen genannten Merkmale umfassen, werden dem Fachmann einfallen.

[0119] Die Erfindung wurde zwar in Hinblick auf eine begrenzte Anzahl von Ausführungsbeispielen beschrieben, doch lassen sich viele Abwandlungen, Modifikationen und sonstige Anwendungen der Erfindung vornehmen.

Zusammenfassung

Verfahren und System für die Umsetzung eines Fast-Wakeup eines Flashspeichersystems

[0120] Verfahren und Systeme für die Pflege von Datenstrukturen auf Grundlage der Ereignisse in einem nicht volatilen Speichersystem. Mindestens ein Teil einer oder mehrerer Managementtabellen sowie eine künftige Informationsdatenstruktur sind in einem nicht volatilen Speicher hinterlegt. Die künftige Informationsdatenstruktur umfasst Datensätze zu Ereignissen, deren Eintritt nach dem Speichern der künftigen Informationsdatenstruktur erwartet wird. Treten Ereignisse im Flashspeicher ein, werden sie auf Grundlage der künftigen Informationsdatenstruktur bearbeitet. Beim Hochfahren des Speichersystems wird/werden die Managementtabelle(n) geladen und die Datensätze der künftigen Informationsdatenstruktur mit dem Status der Tabelle(n) verglichen. Die Tabelle(n) wird/werden auf Grundlage der künftigen Informationsdatenstruktur aktualisiert.

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 Patentliteratur

- US 6510488 **[0002, 0095]**
- US 5937425 **[0004, 0060]**
- US 6678785 **[0004]**

Patentansprüche

1. Speichermodul, gekennzeichnet durch:
(a) einen ersten nicht volatilen Speicher; und
(b) eine Steuerung des genannten ersten nicht volatilen Speichers, die dazu dient, den ersten nicht volatilen Speicher durch Schritte zu steuern, die folgende Schritte umfassen
(i) Speichern im genannten ersten nicht volatilen Speicher mindestens eines Teils einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt abbilden,
(ii) Speichern im genannten ersten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen in Bezug auf Ereignisse im Speichersystem umfasst, deren Eintritt nach dem Speichern der genannten künftigen Informationsdatenstruktur erwartet wird, und
(iii) zu einem zweiten Zeitpunkt, der auf das Speichern der genannten künftigen Informationsdatenstruktur folgt, Bearbeitung des genannten Ereignisses gemäß der künftigen Informationsdatenstruktur.

2. Speichermodul gemäß Anspruch 1, gekennzeichnet durch:
(c) einen zweiten nicht volatilen Speicher;
wobei die genannte Steuerung dazu dient, die genannten Schritte mittels Ausführen von im zweiten nicht volatilen Speicher hinterlegtem Code umzusetzen.

3. Speichersystem, gekennzeichnet durch:
(a) ein Speichermodul einschließlich eines nicht volatilen Speichers; und
(b) einen Hostrechner des genannten Speichermoduls, der bei der Steuerung des nicht volatilen Speichers durch Schritte mitwirkt, die folgende Schritte umfassen:
(i) Speichern im genannten ersten nicht volatilen Speicher mindestens eines Teils einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt abbilden,
(ii) Speichern im genannten ersten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen in Bezug auf Ereignisse im Speichersystem umfasst, deren Eintritt nach dem genannten Speichern der genannten künftigen Informationsdatenstruktur erwartet wird, und
(iii) zu einem zweiten Zeitpunkt nach dem genannten Speichern der künftigen Informationsdatenstruktur Bearbeiten eines Ereignisses auf Grundlage der genannten künftigen Informationsdatenstruktur.

4. Speichersystem gemäß Anspruch 3, dadurch gekennzeichnet, dass die genannten Schritte ausschließlich durch den genannten Hostrechner ausgeführt werden.

5. Speichersystem gemäß Anspruch 3, dadurch gekennzeichnet, dass das genannte Speichermodul eine Steuerung umfasst, die mit dem genannten Hostrechner zusammenwirkt, um die genannten Schritte auszuführen.

6. Speichermodul, gekennzeichnet durch:
(a) einen ersten nicht volatilen Speicher; und
(b) eine Steuerung des nicht volatilen Speichers, die dazu dient, das Speichermodul durch Schritte hochzufahren, die folgende Schritte umfassen
(i) Auslesen aus dem genannten ersten nicht volatilen Speicher mindestens eines Teils einer Managementtabelle, die einen Status des Speichermoduls zu einem Zeitpunkt vor dem Hochfahren abbildet;
(ii) Auslesen aus dem genannten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen in Bezug auf Ereignisse umfasst, deren Eintritt nach dem Speichern der genannten künftigen Informationsdatenstruktur erwartet wird; und
(iii) Aktualisieren der genannten Managementtabelle auf Grundlage mindestens eines Datensatzes der genannten künftigen Informationsdatenstruktur.

7. Das Speichermodul gemäß Anspruch 6, gekennzeichnet durch:
(c) einen zweiten nicht volatilen Speicher;
wobei die genannte Steuerung dazu dient, die genannten Schritte durch das Ausführen von Code durchzuführen, der im genannten zweiten nicht volatilen Speicher hinterlegt ist.

8. Speichersystem, gekennzeichnet durch:
(a) ein Speichermodul, das einen nicht volatilen Speicher umfasst; und
(b) einen Hostrechner des genannten Speichermoduls, der bei der Steuerung des genannten nicht volatilen Speichers durch Schritte mitwirkt, die folgende Schritte umfassen:
(i) Auslesen aus dem genannten nicht volatilen Speicher mindestens eines Teils einer Managementtabelle, die einen Status des Speichersystems zu einem Zeitpunkt vor dem Hochfahren abbildet;
(ii) Auslesen aus dem genannten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen umfasst, die sich auf Ereignisse beziehen, deren Eintritt nach einem Speichern der genannten künftigen Informationsdatenstruktur erwartet wird; und
(iii) Aktualisieren der genannten Managementtabelle auf Grundlage mindestens eines Datensatzes der genannten künftigen Informationsdatenstruktur.

9. Speichersystem gemäß Anspruch 8, dadurch gekennzeichnet, dass die genannten Schritte ausschließlich vom Hostrechner ausgeführt werden.

10. Speichersystem gemäß Anspruch 8, dadurch gekennzeichnet, dass das genannte Speichermodul eine Steuerung umfasst, die mit dem genannten Hostrechner bei der Umsetzung der genannten

Schritte zusammenwirkt.

11. Computerlesbares Speichermedium mit darin integriertem computerlesbaren Code für die Pflege der Datenstrukturen eines Speichersystems auf Grundlage von Systemereignissen, wobei der computerlesbare Code gekennzeichnet ist durch:
(a) Programmcode für das Speichern mindestens eines Teils einer Managementtabelle, deren Inhalte einen Status des Systems zu einem ersten Zeitpunkt abbilden, in einem nicht volatilen Speicher des Speichersystems;
(b) Programmcode für das Speichern einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen umfasst, die sich auf Ereignisse im Speichersystem beziehen, deren Eintritt nach dem genannten Speichern der genannten Informationsdatenstruktur erwartet wird, im genannten nicht volatilen Speicher; und
(c) Programmcode für die Bearbeitung eines Ereignisses auf Grundlage der künftigen Informationsdatenstruktur zu einem zweiten Zeitpunkt nach dem genannten Speichern der genannten künftigen Informationsdatenstruktur.

12. Computerlesbares Speichermedium, gekennzeichnet durch darin integrierten computerlesbaren Code für das Hochfahren eines Speichersystems, wobei der computerlesbare Code Folgendes umfasst:
(a) Programmcode für das Auslesen aus einem nicht volatilen Speicher des Speichersystems mindestens eines Teils einer Managementtabelle, die einen Status des Speichersystems zu einem Zeitpunkt vor dem Hochfahren abbildet;
(b) Programmcode für das Auslesen aus dem genannten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen umfasst, die sich auf Ereignisse beziehen, deren Eintritt nach einem Speichern der genannten künftigen Informationsdatenstruktur erwartet wird; und
(c) Programmcode für das Aktualisieren der genannten Managementtabelle auf Grundlage mindestens eines Datensatzes der genannten künftigen Informationsdatenstruktur.

13. Verfahren für die Pflege von Datenstrukturen eines Speichersystems auf Grundlage der Ereignisse im System, gekennzeichnet durch folgende Schritte:
(a) Speichern in einem nicht volatilen Speicher des Speichersystems mindestens eines Teils einer Managementtabelle, deren Inhalte einen Status des Speichersystems zu einem ersten Zeitpunkt abbilden;
(b) Speichern im genannten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen umfasst, die sich auf Ereignisse im Speichersystem beziehen, deren Eintritt nach dem genannten Speichern der genannten Informationsdatenstruktur erwartet wird; und
(c) zu einem zweiten Zeitpunkt nach dem genannten Speichern der künftigen Informationsdatenstruktur Bearbeitung eines Ereignisses auf Grundlage der genannten künftigen Informationsdatenstruktur.

14. Verfahren gemäß Anspruch 13, dadurch gekennzeichnet, dass der genannte nicht volatile Speicher ein Flashspeicher ist.

15. Verfahren gemäß Anspruch 13, dadurch gekennzeichnet, dass das genannte Speichern des genannten mindestens einen Teils der genannten Managementtabelle das Aktualisieren des genannten mindestens einen Teils der genannten Managementtabelle im genannten nicht volatilen Speicher umfasst.

16. Verfahren gemäß Anspruch 15, dadurch gekennzeichnet, dass das genannte Speichern des genannten mindestens einen Teils der Managementtabelle, deren Inhalte einen Zustand des Speichersystem zu einem ersten Zeitpunkt abbilden, auch das Speichern mindestens eines Teils der genannten Managementtabelle in einem volatilen Speicher umfasst, sowie dadurch gekennzeichnet, dass das genannte Aktualisieren jedes N > 1 Mal erfolgt, das die Managementtabelle im genannten volatilen Speicher geändert wird.

17. Verfahren gemäß Anspruch 15, dadurch gekennzeichnet, dass das genannte Aktualisieren in Reaktion auf den Eintritt ausgewählter Ereignisse folgt.

18. Verfahren gemäß Anspruch 15, dadurch gekennzeichnet, dass das genannte Aktualisieren periodisch erfolgt.

19. Verfahren gemäß Anspruch 15, dadurch gekennzeichnet, dass das genannte Aktualisieren auf Grundlage der Verfügbarkeit von Kapazität im Speichersystem erfolgt.

20. Verfahren gemäß Anspruch 13, dadurch gekennzeichnet, dass die genannte künftige Informationsdatenstruktur eine Auflistung von Blöcken des genannten nicht volatilen Speichers umfasst, die zum genannten ersten Zeitpunkt unbelegt sind.

21. Verfahren zum Hochfahren eines Speichersystems, gekennzeichnet durch folgende Schritte:
(a) Auslesen aus einem nicht volatilen Speicher des Speichersystems mindestens eines Teils einer Managementtabelle, die einen Status des Speichersystems zu einem Zeitpunkt vor dem Hochfahren abbildet;
(b) Auslesen aus dem genannten nicht volatilen Speicher einer künftigen Informationsdatenstruktur, die eine Mehrzahl von Datensätzen umfasst, die sich auf

Ereignisse beziehen, deren Eintritt nach einem Speichern der genannten künftigen Informationsdatenstruktur erwartet wird; und
(c) Aktualisieren der genannten Managementtabelle auf Grundlage mindestens eines Datensatzes der genannten künftigen Informationsdatenstruktur.

22. Verfahren gemäß Anspruch 21, dadurch gekennzeichnet, dass der genannte nicht volatile Speicher ein Flashspeicher ist.

23. Verfahren gemäß Anspruch 21, dadurch gekennzeichnet, dass das genannte Aktualisieren die genannte Managementtabelle ändert, um einen aktuellen Status des Speichersystems abzubilden.

24. Verfahren gemäß Anspruch 21, dadurch gekennzeichnet, dass das genannte Aktualisieren bedingt erfolgt.

25. Verfahren gemäß Anspruch 24, gekennzeichnet durch folgenden weiteren Schritt:
(d) Prüfung der genannten Datensätze, um zu bestimmen, ob der Status des Speichersystems sich seit dem genannten Zeitpunkt geändert hat,
wobei das Aktualisieren voraussetzt, dass der Status des Speichersystems sich seit diesem Zeitpunkt geändert hat.

Es folgen 7 Blatt Zeichnungen

Anhängende Zeichnungen

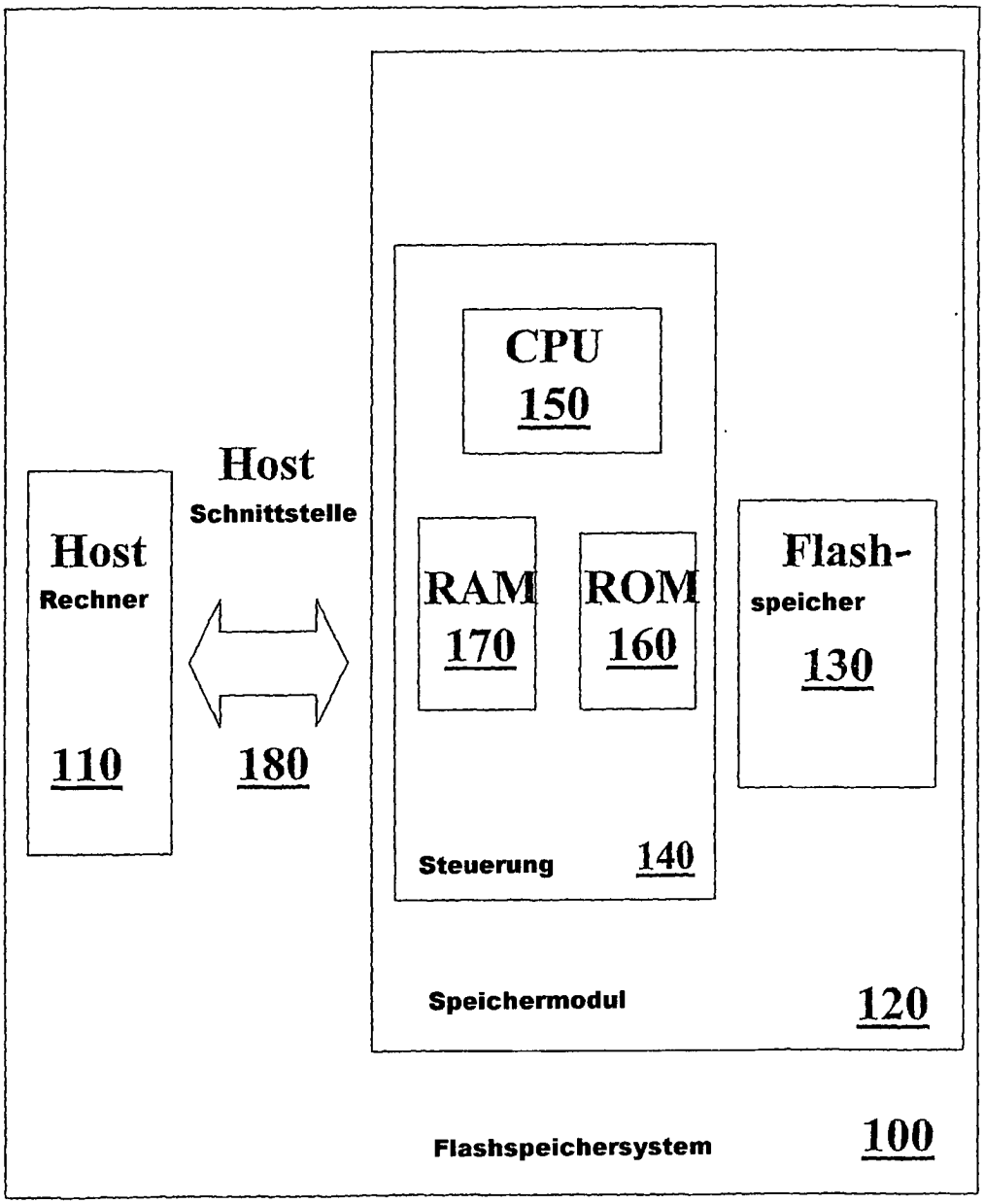


FIG.1

Für $t=t_1$ Translation **stabelle** **200**

Virtueller Block	**Physischer** Block
0	893
1	485
2	**172**
•	•
•	•
•	•
1022	1
1023	985

FIG.2A

Für $t=t_2$ wenn $t_2>t_1$ Translationstabelle **200**

Virtueller Block	Physischer Block
0	893
1	485
2	777
.	.
.	.
.	.
1022	1
1023	985

FIG.2B

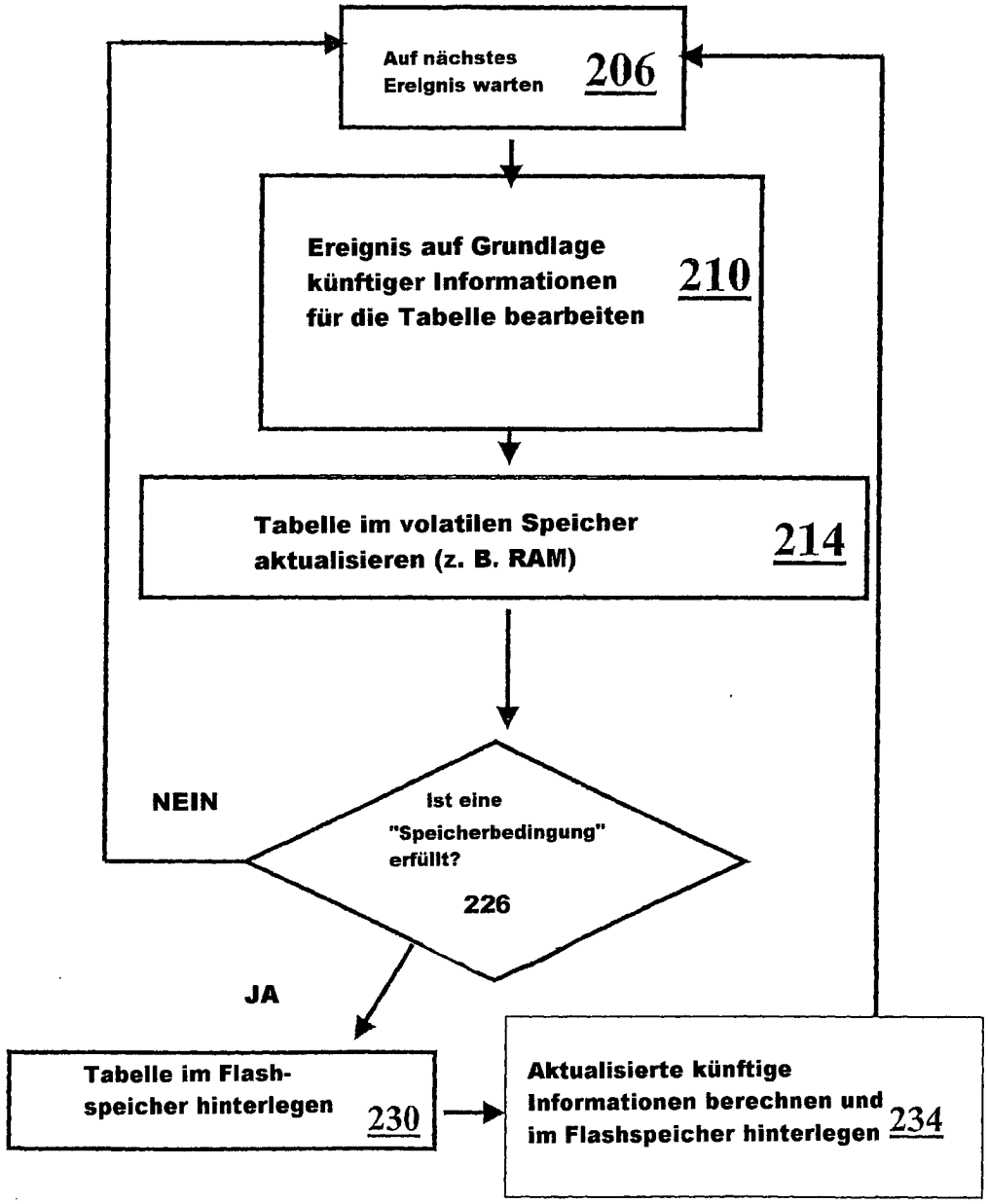

Auf nächstes Ereignis warten
206
Ereignis auf Grundlage künftiger Informationen für die Tabelle bearbeiten
210
Tabelle im volatilen Speicher aktualisieren (z. B. RAM)
214
NEIN
Ist eine "Speicherbedingung" erfüllt?
226
JA
Tabelle im Flash-speicher hinterlegen
230
Aktualisierte künftige Informationen berechnen und im Flashspeicher hinterlegen
234

FIG.3

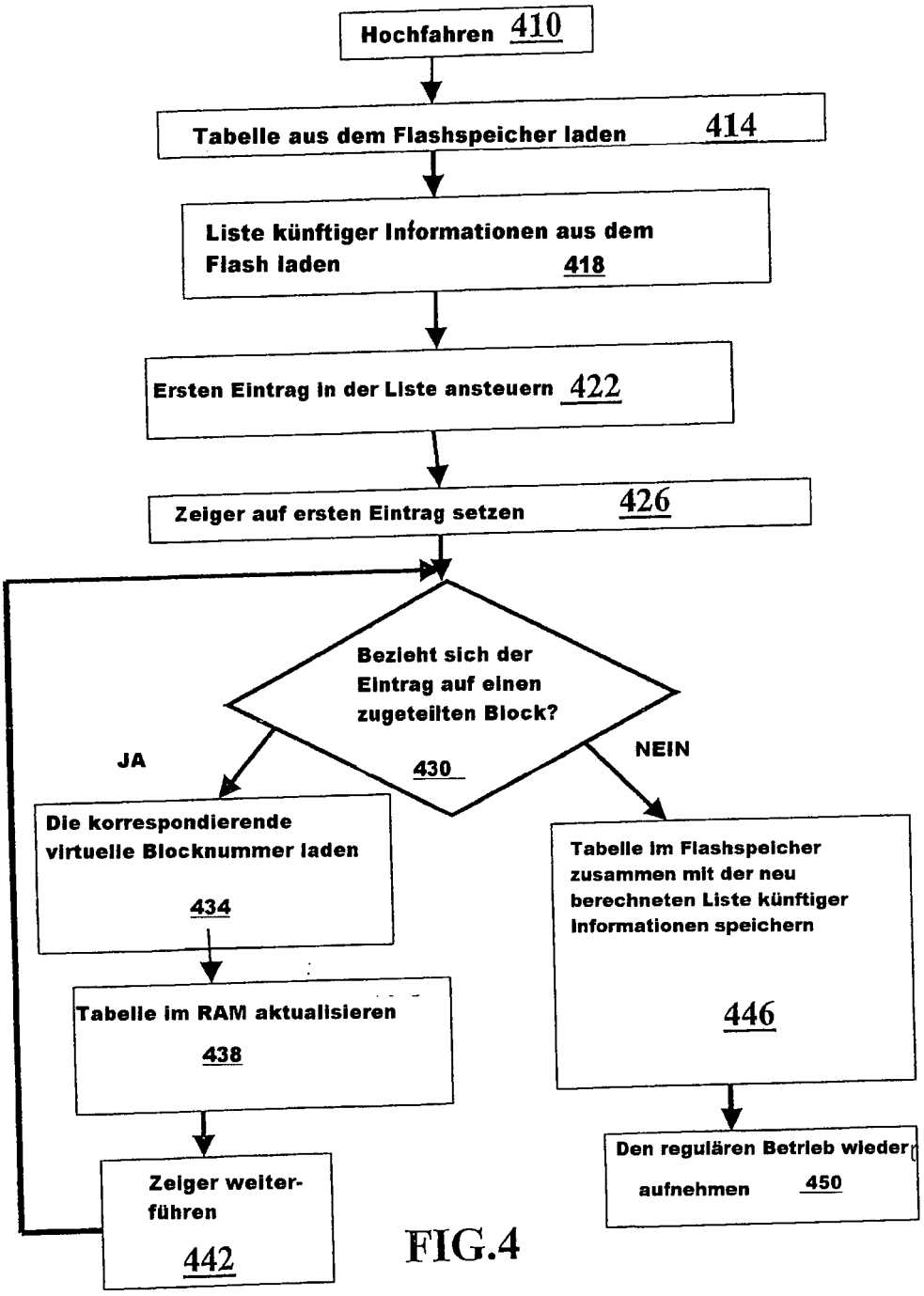

Hochfahren 410
Tabelle aus dem Flashspeicher laden 414
Liste künftiger Informationen aus dem Flash laden 418
Ersten Eintrag in der Liste ansteuern 422
Zeiger auf ersten Eintrag setzen 426
Bezieht sich der Eintrag auf einen zugeteilten Block? 430
JA
NEIN
Die korrespondierende virtuelle Blocknummer laden 434
Tabelle im RAM aktualisieren 438
Zeiger weiter-führen 442
Tabelle im Flashspeicher zusammen mit der neu berechneten Liste künftiger Informationen speichern 446
Den regulären Betrieb wieder aufnehmen 450

FIG.4

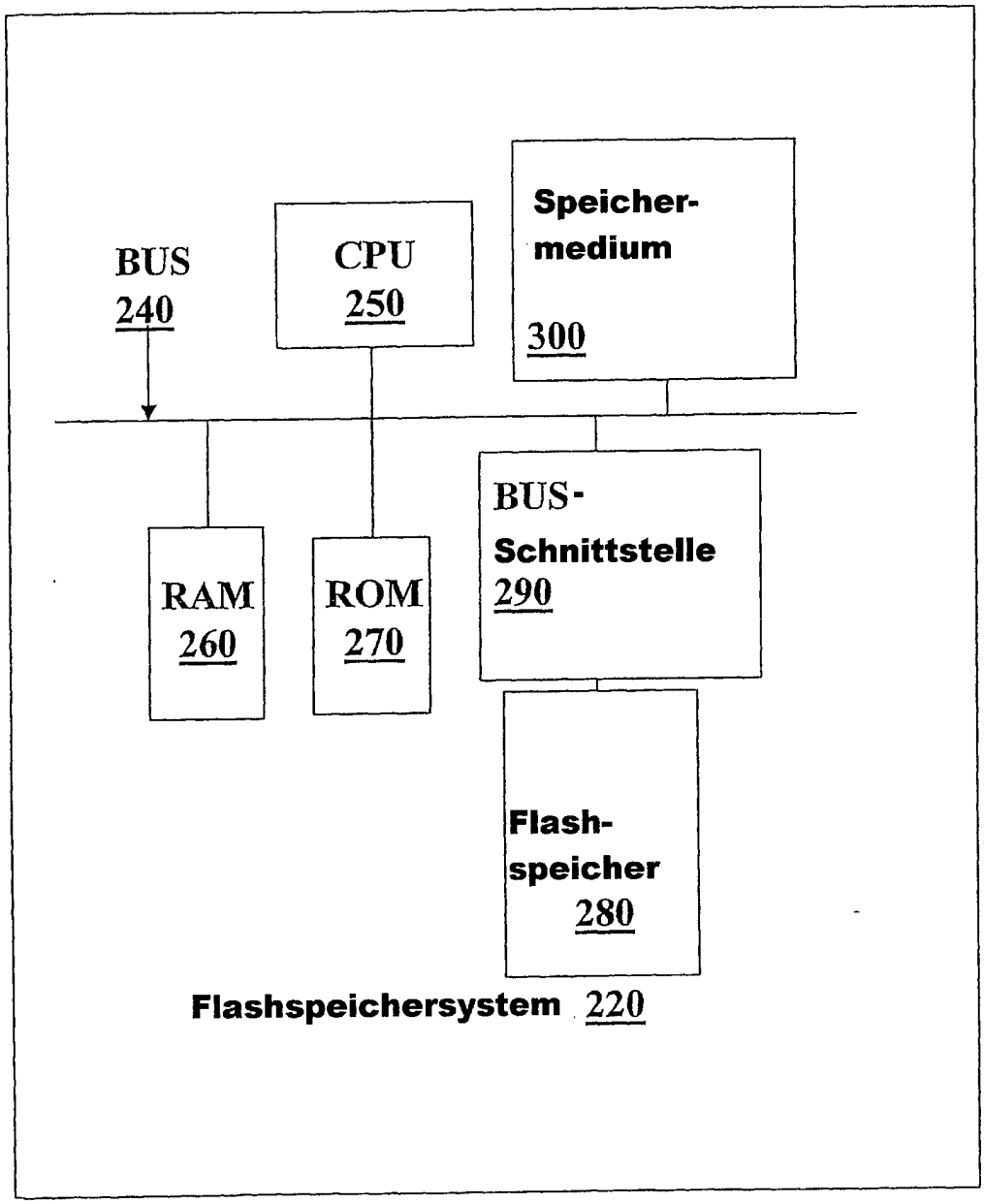

BUS
240
CPU
250
Speicher-
medium
300
BUS-
Schnittstelle
290
RAM
260
ROM
270
Flash-
speicher
280
Flashspeichersystem 220

FIG.5

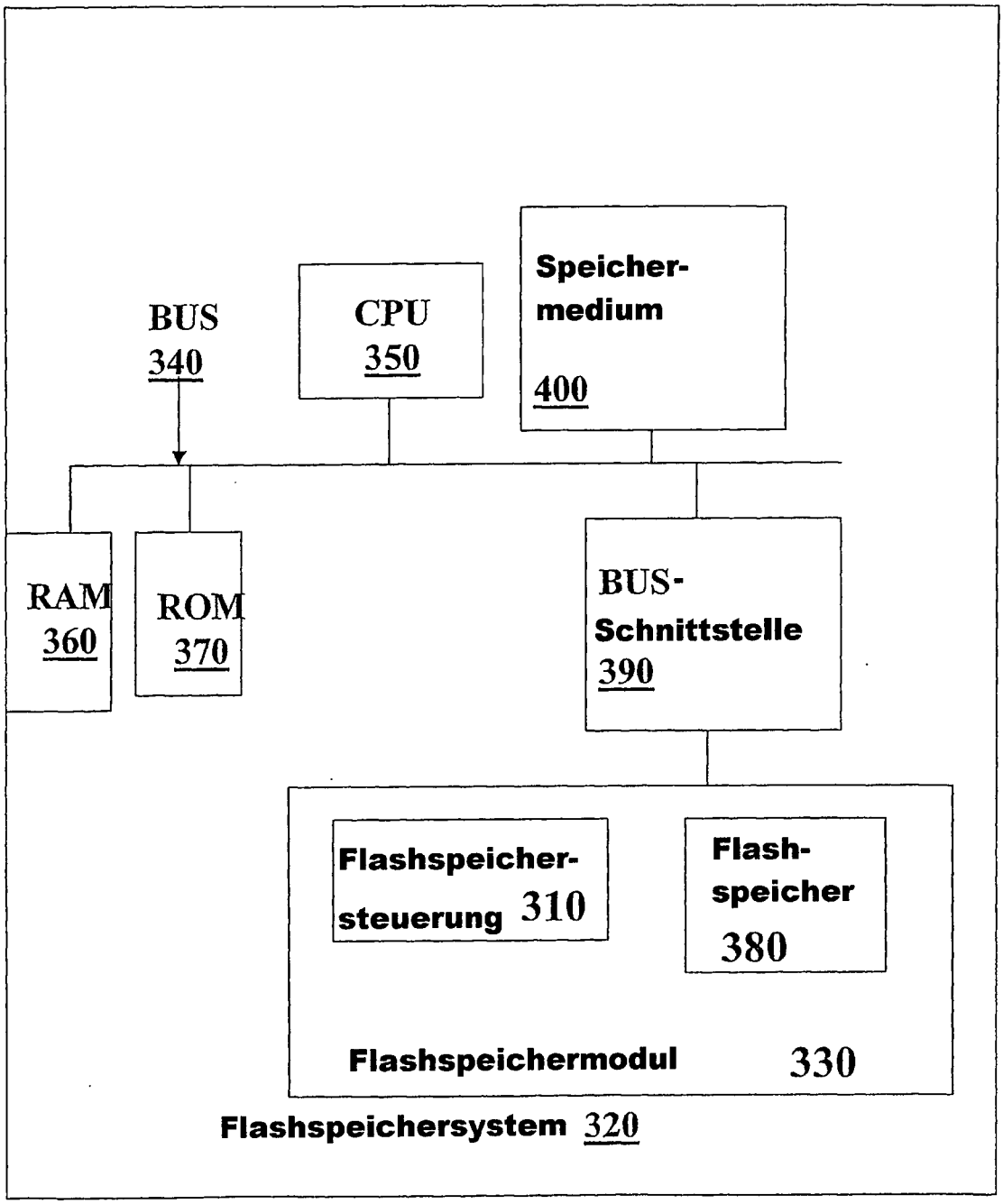

BUS
340
CPU
350
Speicher-
medium
400
RAM
360
ROM
370
BUS-
Schnittstelle
390
Flashspeicher-
steuerung 310
Flash-
speicher
380
Flashspeichermodul 330
Flashspeichersystem 320

FIG.6