(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 546 562**

(51) Int. CI.:

G06F 17/30 (2006.01)
G06F 11/14 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **25.06.2010 E 10730310 (9)**

(97) Fecha y número de publicación de la concesión europea: **12.08.2015 EP 2433226**

(54) Título: **Sistema de archivos**

(30) Prioridad:

26.06.2009 US 269633 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
24.09.2015

(73) Titular/es:

SIMPLIVITY CORPORATION (100.0%)
8 Technology Drive
Westborough, MA 01581-1756, US

(72) Inventor/es:

BEAVERSON, ARTHUR J. y
BOWDEN, PAUL

(74) Agente/Representante:

UNGRÍA LÓPEZ, Javier

DESCRIPCIÓN

Sistema de archivos

Campo de la invención

La presente invención se refiere a estructuras de datos de sistema de archivos informáticos y a métodos y aparatos para la denominación y el almacenamiento de archivos.

Antecedentes

Una solución de almacenamiento con todas las funciones incluidas puede incluir discos en bruto, un sistema de archivos, instantáneas, control de versiones de archivos, compresión, cifrado, optimización de capacidades incorporadas (por ejemplo, desduplicación de datos), otras características de seguridad tales como auditoría y resistencia a manipulación indebida, una replicación eficiente en una ubicación fuera del sitio con fines de recuperación en casos de desastre, y así sucesivamente. Muchas de estas características se entregan en aparatos separados que entonces han de ser conectados por unos técnicos sumamente experimentados.

La construcción de una solución de almacenamiento de este tipo con la tecnología de hoy en día, para muchos terabytes (TB) de datos, a menudo da como resultado una solución multicaja que puede superar con facilidad unos costes de 100.000 dólares, lo que hace que una solución de almacenamiento con todas las funciones incluidas de este tipo no se encuentre disponible para muchas empresas y clientes.

Esta solución *ad hoc* multicaja no es un aspecto fundamental del almacenamiento, sino más bien que las implementaciones y arquitecturas de sistema de archivos no se han mantenido a la par con otros desarrollos tecnológicos. Por ejemplo, la mayor parte de las arquitecturas de sistema de archivos no han evolucionado para aprovechar completamente las unidades de procesamiento informático (CPU, *computer processing unit*), más rápidas, la memoria flash, y el diferente equilibrio entre el ancho de banda de red, la densidad de disco y las tasas de acceso a disco.

Si se define la accesibilidad de los datos como la relación del ancho de banda de acceso con respecto al almacenamiento direccionable, la accesibilidad de los datos está disminuyendo. Las densidades de almacenamiento están aumentando más rápido que el acceso a los discos, por lo tanto para un tamaño de conjunto de datos dado, el tiempo que se necesita para acceder a los datos está aumentando (y por lo tanto, dando lugar a una accesibilidad reducida). El efecto sobre las arquitecturas de almacenamiento es tal como sigue: una vez que se almacenan los datos, no deberían moverse a menos que sea absolutamente necesario. Esta simple observación se infringe muchas veces en las arquitecturas de almacenamiento actuales donde los datos constantemente se están colocando en memoria y escribiéndose de nuevo. El resultado es un gasto adicional significativo (por ejemplo, 10 canales, CPU, potencia, tiempo, gestión).

Se hace referencia al documento US 2009/0037456 que describe un almacén de datos donde se almacenan fragmentos. Cada recopilación de fragmentos forma unos archivos respectivos. Se proporciona un índice que pone en correspondencia compendios de fragmentos con páginas que contienen información para recrear los fragmentos, en donde el índice se almacena en un almacenamiento persistente.

Sumario de la invención

La invención se expone en la reivindicación independiente 1.

De acuerdo con un ejemplo de la invención, se proporciona un sistema de archivos que comprende:

un sistema de archivos digitalmente firmado donde datos, metadatos y archivos son objetos, teniendo cada objeto una huella digital globalmente única y derivada de contenido y donde las referencias de objeto se ponen en correspondencia mediante las huellas digitales;
teniendo el sistema de archivos un objeto raíz que comprende una puesta en correspondencia de todas las huellas digitales de objeto en el sistema de archivos;
en el que un cambio en el sistema de archivos da como resultado un cambio en el objeto raíz, y realizar un seguimiento de los cambios en el objeto raíz proporciona una historia de la actividad de sistema de archivos.

En un ejemplo:

el sistema de archivos incluye un objeto de correspondencia de nodos - i que comprende una puesta en correspondencia de números de nodo - i con huellas digitales de objeto de archivo y donde la huella digital del objeto de correspondencia de nodos - i comprende una instantánea del sistema de archivos.

De acuerdo con otro ejemplo de la invención, se proporciona un medio legible por ordenador que contiene unas instrucciones de programa ejecutables para un método indexación de objetos almacenados, comprendiendo el método:

proporcionar datos, metadatos y archivos como objetos;
proporcionar una huella digital para cada objeto que es globalmente única y que se deriva del contenido del objeto; y
en el que se proporciona un objeto raíz de sistema de archivos que comprende una puesta en correspondencia de todas las huellas digitales de objeto en el sistema de archivos, de tal modo que un cambio en el sistema de archivos da como resultado un cambio en el objeto raíz, y realizar un seguimiento de los cambios en el objeto raíz proporciona una historia de la actividad de sistema de archivos.

En un ejemplo, el método incluye:

proporcionar un objeto de correspondencia de nodos - i de sistema de archivos que comprende una puesta en correspondencia de números de nodo - i con huellas digitales de objeto de archivo,
en el que la huella digital del objeto de correspondencia de nodos - i comprende una instantánea del sistema de archivos.

En un ejemplo, el método incluye:

publicar la huella digital de correspondencia de nodos - i en otro sistema informático en un almacén de objetos distinto.

En un ejemplo, el método incluye:

usar la huella digital de correspondencia de nodos - i como una instantánea del sistema de archivos para una recuperación en casos de desastre.

En un ejemplo:

el objeto de correspondencia de nodos - i contiene una huella digital de una correspondencia de nodos - i previa.

En un ejemplo:

las huellas digitales de correspondencia de nodos - i previa comprenden una historia de instantáneas del sistema de archivos.

En un ejemplo:

los objetos tienen unos recuentos de referencia; y
tras un cambio en el sistema de archivos, ajustar los recuentos de referencia de objeto de cada objeto por debajo del objeto de correspondencia de nodos - i.

En un ejemplo:

el ajuste se realiza en cada transacción de E / S para proporcionar una protección de datos continua.

En un ejemplo:

el ajuste se realiza de forma periódica, a demanda, o al ocurrir sucesos particulares para generar unas instantáneas.

En un ejemplo:

los objetos tienen unos recuentos de referencia; y
los ajustes a los recuentos de referencia se utilizan para la desduplicación de datos de tal modo que solo se almacena un nuevo contenido de datos.

De acuerdo con otro ejemplo de la invención, se proporciona un sistema de archivos de ordenador para la denominación y el almacenamiento de archivos en uno o más dispositivos de almacenamiento en ordenador, comprendiendo el sistema:

un sistema de archivos de espacio de nombres donde archivos, datos y metadatos son objetos, teniendo cada objeto una huella digital globalmente única que se deriva del contenido del objeto, comprendiendo cada objeto de archivo una puesta en correspondencia de huellas digitales de objeto para los objetos de datos y / u objetos de

metadatos del archivo y teniendo el objeto de archivo su propia huella digital de objeto derivada de las huellas digitales de los objetos en el archivo, y donde el sistema incluye una puesta en correspondencia de números de nodo - i con las huellas digitales de objeto de archivo.

En un ejemplo:

las referencias de objeto se definen mediante las huellas digitales de objeto.

En un ejemplo:

la puesta en correspondencia de objetos de archivo comprende una lista lineal, una estructura en árbol o una tabla de direccionamiento indirecto.

En un ejemplo:

los objetos de archivo incluyen un objeto raíz que tiene su propia huella digital de objeto derivada de la totalidad de los objetos en el sistema de archivos de tal modo que puede accederse a cada objeto en el sistema de archivos a través del objeto raíz.

En un ejemplo:

el sistema de archivos de espacio de nombres se proporciona como una capa en una pila de almacenamiento entre una capa de sistema de archivos virtual y una capa de abstracción de almacenamiento de bloques.

En un ejemplo, el sistema comprende además:

un almacén de objetos que contiene un índice de huellas digitales de objeto, ubicaciones de objeto y recuentos de referencia de objeto.

En un ejemplo:

el índice de almacén de objetos se almacena en memoria no volátil.

En un ejemplo:

la huella digital es un compendio de troceo criptográfico del contenido de objeto.

En un ejemplo:

el tamaño de objeto es variable.

En un ejemplo:

el sistema de archivos es un sistema de archivos conforme con POSIX.

De acuerdo con otro ejemplo de la invención, se proporciona un método que comprende:

generar unas huellas digitales de objeto para unos objetos de datos en un sistema de archivos, comprendiendo los objetos de datos unos datos o metadatos, y comprendiendo las huellas digitales de objeto una huella digital globalmente única que se deriva del contenido de objeto de datos;
generar unas huellas digitales de objeto para objetos de archivo, donde cada objeto de archivo comprende las huellas digitales de una pluralidad de los objetos de datos en el archivo y la huella digital de objeto de archivo comprende una huella digital globalmente única que se deriva del contenido de objeto de archivo; y
generar un objeto raíz que comprende una puesta en correspondencia de la totalidad de las huellas digitales de objeto en el sistema de archivos.

En un ejemplo, el método comprende:

mantener un recuento de referencia para cada objeto, y actualizar el recuento de referencia del objeto cuando se añaden o se eliminan referencias al objeto.

En un ejemplo, el método comprende:

generar un registro de transacciones de actividad de objetos, incluyendo lecturas, escrituras, eliminaciones y actualizaciones de recuentos de referencia.

En un ejemplo, el método comprende:

añadir, modificar o eliminar un objeto de datos en un archivo y generar una nueva huella digital de objeto de archivo.

En un ejemplo:

cuando se cambia el contenido de un objeto de archivo o un objeto de datos, propagar el cambio hacia arriba hasta el objeto raíz.

En un ejemplo, el método comprende:

realizar la etapa de propagación en uno de:

cada transacción de E / S;
de forma periódica;
a demanda;
en un suceso particular.

De acuerdo con otro ejemplo de la invención, se proporciona un método que comprende:

proporcionar una pluralidad de objetos de datos, comprendiendo cada objeto de datos unos datos o metadatos, y teniendo cada objeto de datos una huella digital que es globalmente única y que se deriva de su contenido; y
generar un objeto de archivo que comprende una pluralidad de huellas digitales de objeto de datos para una pluralidad de objetos de datos asociados, y generar una huella digital de objeto de archivo que es globalmente única y que se deriva del contenido del objeto de archivo; y
mantener un índice de números de nodo - i para huellas digitales de objeto de archivo.

En un ejemplo, el método comprende:

mantener un índice de ubicaciones para poner en correspondencia huellas digitales de objeto y ubicaciones físicas de los objetos.

En un ejemplo:

el índice de ubicaciones incluye unos recuentos de referencia para los objetos.

En un ejemplo:

las huellas digitales y los índices comprenden un sistema de archivos.

De acuerdo con otro ejemplo de la invención, se proporciona:

un medio legible por ordenador que contiene unas instrucciones de programa ejecutables para un método indexación de objetos almacenados, comprendiendo el método:

generar unas huellas digitales que son globalmente únicas y que se derivan del contenido de objetos de datos y de metadatos;
generar unos objetos de archivo que comprenden una pluralidad de huellas digitales de objetos de datos y / o de metadatos y generar unas huellas digitales de los objetos de archivo que son globalmente únicas y que se derivan para el contenido del objeto de archivo; y
generar un objeto raíz que comprende una puesta en correspondencia de la totalidad de las huellas digitales de los objetos de datos, de metadatos y de archivo.

De acuerdo con otro ejemplo de la invención, se proporciona:

un procesador físico y unos dispositivos de almacenamiento que proporcionan un acceso a datos, metadatos y archivos; y
en el que los datos, los metadatos y los archivos son objetos, teniendo cada objeto una huella digital globalmente única y derivada de en contenido y donde las referencias de objeto están indexadas por las huellas digitales; y
la indexación incluye una puesta en correspondencia de números de nodo - i con las huellas digitales de objeto de archivo.

De acuerdo con otro ejemplo de la invención, se proporciona:

un aparato de procesamiento y de almacenamiento para denominar y almacenar objetos de datos y recopilaciones de objetos de datos que comprenden objetos de archivo, comprendiendo cada objeto de datos unos datos o metadatos y teniendo cada objeto una huella digital globalmente única basada en contenido como su nombre de objeto, siendo el objeto de archivo una recopilación de nombres de objeto de datos y teniendo su propia huella digital globalmente única basada en contenido como su nombre de objeto de archivo;
un sistema de archivos que tiene dos capas que incluye:

una capa de almacén de objetos que incluye un índice de puesta en correspondencia de nombres de objeto y ubicaciones físicas de objeto; y
una capa de espacio de nombres que incluye un índice de puesta en correspondencia de nombres de objeto de datos para cada objeto de archivo.

En un ejemplo:

la capa de espacio de nombres incluye un índice de puesta en correspondencia de números de nodo - i con los nombres de objeto de archivo.

En un ejemplo:

la capa de almacén de objetos incluye unos recuentos de referencia para cada objeto.

En un ejemplo:

el nombre de objeto es un compendio de troceo criptográfico del contenido de objeto.

En un ejemplo:

el sistema incluye un aparato de aceleración de soporte físico para realizar uno o más de denominación de objetos, compresión y cifrado.

En un ejemplo:

la capa de almacén de objetos incluye un índice global de todos los objetos en el sistema de archivos, donde la clave primaria para el índice de objetos global es el nombre de objeto, y el nombre de objeto es un compendio de troceo criptográfico del contenido de objeto.

Breve descripción de los dibujos

La invención se entenderá más completamente por referencia a la descripción detallada, junto con las siguientes figuras, donde:

la figura 1 es un diagrama de bloques esquemático que ilustra una realización de la invención integrada en un espacio de núcleo de sistema operativo;

la figura 2 es un diagrama de bloques esquemático de los componentes principales de una realización de un almacén de objetos, posibilitando que el almacén de objetos se hospede en una diversidad de medios físicos;

la figura 3 es un diagrama de bloques esquemático de una realización de un almacén de objetos que puede extraer por abstracción una funcionalidad clave, posibilitando que dicha funcionalidad se implemente en una diversidad de formas sin afectar al diseño de almacén de objetos; las implementaciones pueden variar de una solución de soporte lógico puro, a una que usa una aceleración de soporte físico;

la figura 4 es un diagrama de bloques esquemático de una realización de un conjunto de objetos que están agrupados de forma conjunta en una construcción (“nodo - h”) como un bloque de construcción básico de un sistema de archivos integrado;

la figura 5 es un diagrama de bloques esquemático de una realización de un nodo - h que puede especializarse en otras estructuras de datos según lo necesite el sistema de archivos, tal como archivos, directorios y correspondencias - i;

la figura 6 es un diagrama de bloques esquemático de una realización que ilustra cómo los cambios en el sistema de archivos son objeto de seguimiento y se mantienen con el tiempo, y cómo las técnicas que se usan dan como resultado, de forma natural, eficiencia espacial, inmutabilidad y seguridad;

la figura 7 es un diagrama de bloques esquemático de una realización de un objeto que puede manejar de forma transparente la compresión, el cifrado y la independencia de la ubicación a la vez que se proporciona un nombre globalmente único para el objeto; y

la figura 8 es un diagrama de bloques esquemático de una realización alternativa de la invención que se implementa en el espacio de usuario con FUSE (*File System in User Space*, sistema de archivos en espacio de usuario); FUSE es un conjunto de código abierto de librerías y módulos de núcleo que posibilitan la construcción de sistemas de archivos en el espacio de usuario;

la figura 9 es un diagrama de bloques esquemático que ilustra diversas operaciones de indexación que se realizan de acuerdo con una realización de una invención de indexación;

las figuras 10A a 10D ilustran diversas realizaciones de estructuras de datos que pueden usarse en la invención;

la figura 11 es un diagrama de bloques esquemático que ilustra una operación de consulta de acuerdo con una realización de la invención;

la figura 12 es un diagrama de bloques esquemático que ilustra una operación de inserción de acuerdo con una realización de la invención;

la figura 13 es un diagrama de bloques esquemático de una operación de eliminación de acuerdo con una realización de la invención;

la figura 14 es un diagrama de bloques esquemático de una operación de actualización de acuerdo con una realización de la invención;

las figuras 15A y 15B son unos diagramas de bloques esquemáticos que ilustran un proceso de lectura aleatoria para generar unos bloques de borrado libres de acuerdo con una realización de la invención;

las figuras 16A y 16B son unos diagramas de bloques esquemáticos que ilustran otro método de generación de unos bloques de borrado libres de acuerdo con un proceso de depuración;

la figura 17 es un diagrama de bloques esquemático que ilustra una vista o pila de seis capas para ilustrar una implementación de la invención;

la figura 18 es un diagrama esquemático de una entrada de registro tal como se usa en una realización de la invención;

las figuras 19A - 19E ilustran de forma esquemática una implementación de aplicación de troceo de cuco de acuerdo con una realización de la invención;

la figura 20 es una ilustración esquemática de múltiples sectores de almacenamiento; conteniendo cada sector de almacenamiento múltiples registros de acuerdo con una realización de la invención;

la figura 21 es un diagrama esquemático de los contenidos de un sector de almacenamiento de acuerdo con una realización de la invención;

la figura 22 es un diagrama de bloques esquemático que ilustra un ejemplo de un chip de flash físico que tiene múltiples pastillas, bloques de borrado, páginas y sectores de almacenamiento de acuerdo con una realización de la invención; y

las figuras 23A - 23B ilustran determinados componentes de una capa de gestión de dispositivo de acuerdo con una realización de la invención.

Descripción detallada

A. Estructuras de datos de sistema de archivos tradicionales y limitaciones de los sistemas de archivos (heredados) de la técnica anterior.

Un sistema de archivos tradicional tiene varias estructuras de datos básicas. Además de directorios y archivos visibles por el usuario, las estructuras internas incluyen superbloques, nodos - i, correspondencias de asignación, y registros de transacciones.

Las correspondencias de asignación son unas estructuras de datos que indican qué bloques en un disco se encuentran en uso o no. Estas estructuras de datos pueden ser tan simples como una correspondencia de bits, o tan complicadas como un árbol-B. Las correspondencias de asignación pueden ser grandes, y casi nunca caben en memoria. Una asignación no experimentada de nuevos bloques da como resultado un rendimiento de disco bajo, pero una colocación óptima requiere unos algoritmos de asignación sofisticados dadas las limitaciones de memoria que se han mencionado en lo que antecede.

Los directorios son listas de nombres de archivos y otros directorios, y en muchos sistemas de archivos, se tratan como otro tipo de archivo que simplemente se interpreta de forma diferente. Internamente, un directorio es una lista de pares de nombre de archivo / número de nodo - i. Cuando el sistema de archivos quiere un acceso a un nombre de archivo, este ha de encontrar el nombre de archivo en un directorio, y el número de nodo - i correspondiente.

Los archivos son unas recopilaciones con nombre de datos. Un nombre de archivo, junto con el nodo - i al que este hace referencia, se almacena en una estructura de directorios. Muchos sistemas de archivos soportan el concepto de enlaces, donde diferentes nombres de archivo pueden apuntar a los mismos datos (nodo - i).

Los registros de transacciones se usan para mantener el sistema de archivos consistente de acuerdo con unas propiedades Atómicas, Consistentes, Independientes y Duraderas (ACID, *Atomic, Consistent, Independent and Durable*). Muchos sistemas de archivos garantizarán la consistencia de los metadatos, pero tienen diferentes acuerdos de nivel de servicio (SLA, *service level agreement*) para datos.

Un superbloque es una pequeña estructura de datos que reside en una ubicación conocida en un disco o medio persistente. A partir del superbloque, pueden encontrarse todas las otras estructuras de datos relevantes para el sistema de archivos, tal como el tamaño y la ubicación de la tabla de nodos - i, las correspondencias de asignación, el directorio raíz, y así sucesivamente. Cuando un sistema de archivos está montado, este es el superbloque al que se accede en primer lugar. Por razones de seguridad, los superbloques a menudo se replican en diversos puntos en un disco.

Quizá la estructura de datos más fundamental sea el nodo - i ("nodo de índice"). Común a muchos sistemas de archivos, esta es una estructura de datos que es el contenedor básico para el contenido, tal como un archivo. El propio nodo - i no contiene un nombre de archivo; ese se almacena en el directorio. Un nodo - i se identifica mediante un número entero que indica un índice en una estructura de datos residente en disco (la tabla de nodos - i). Cada entrada de nodo - i en la tabla describe en dónde puede encontrarse en el disco el contenido para este archivo. Esta "correspondencia" puede adoptar diversas formas, incluyendo listas lineales, tablas de direccionamiento indirecto, diversos tipos de árbol, cada uno de los cuales tiene diversas compensaciones recíprocas de velocidad / espacio. Es importante que la correspondencia use un direccionamiento físico o lógico, tal como un número de bloque lógico (LBN, *logical block number*). Un LBN solo cobra sentido si se sabe a qué disco está destinado.

A partir de la descripción anterior, debería ser evidente que los sistemas de archivos heredados tienen un control estricto del qué (el contenido) y el dónde (la colocación de datos). Esta combinación del qué y el dónde, en gran medida un producto secundario de la historia, da como resultado una arquitectura que es difícil de extender a las necesidades de almacenamiento modernas.

B. Estructuras de datos y características novedosas de los sistemas de archivos de la invención.

De acuerdo con diversas realizaciones de la invención, se proporcionan nuevas estructuras de datos para implantar un nuevo tipo de sistema de archivos. El sistema de archivos puede existir y trabajar junto a otros sistemas de archivos; este es compatible con sistemas de archivos heredados y utilidades de nivel de usuario conocidas. No obstante, las nuevas estructuras de datos de la presente invención proporcionan unos beneficios inalcanzables con los sistemas de archivos heredados. Estos beneficios incluyen, pero no se limitan a, uno o más de lo siguiente:

- proporcionar un nivel de abstracción para la denominación y el almacenamiento de archivos que no se basa en un direccionamiento de bloques físicos o lógicos;
- utilizar una huella digital globalmente única que se deriva del contenido de un objeto de datos como el nombre de objeto, comprendiendo cada objeto de datos unos datos o metadatos;
- utilizar huellas digitales de objeto en un sistema de archivos de espacio de nombres donde todas las estructuras de datos internas son objetos, lo que posibilita que todas las referencias entre objetos se definan mediante las huellas digitales de objeto;
- proporcionar una nueva estructura de datos a la que se hace referencia como una estructura de "nodo - h"; un nodo - h de archivo es una estructura de puesta en correspondencia de todas las huellas digitales de objeto de datos en el archivo y él mismo es un objeto que tiene una huella digital globalmente única que se deriva del contenido del objeto de archivo;
- de forma similar, un nodo - h raíz (objeto) es una estructura de puesta en correspondencia de todas las huellas digitales de objeto en el sistema de archivos, de tal modo que cualquier cambio en el sistema de archivos da como resultado un cambio en el objeto raíz y realizar un seguimiento de los cambios al objeto raíz proporciona una historia de la actividad de sistema de archivos;
- proporcionar un objeto de correspondencia de nodos - i (una correspondencia - i) que comprende una puesta en correspondencia de números de nodo - i con huellas digitales de objeto de archivo, posibilitando que el número de nodo - i permanezca constante, mientras que el nombre de objeto (la huella digital) cambia a medida que cambia el contenido del archivo, y donde la huella digital del objeto de correspondencia de nodos - i comprende una instantánea del sistema de archivos.

En las realizaciones que se divulgan, el nombre de objeto, es decir, la huella digital de objeto, es un compendio de troceo criptográfico del contenido del objeto. Esto posibilita que el nombre de objeto sea globalmente único e identificable como una huella digital del contenido de objeto. Una huella digital es significativamente más pequeña que un objeto, por ejemplo, un factor de 100 X, 1000 X o más, y por lo tanto manipular las huellas digitales es con frecuencia más rápido y más fácil que manipular los contenidos subyacentes.

Mediante la provisión de combinaciones o recopilaciones de objetos de datos como nodos - h, que también son objetos que tienen un nombre de objeto que es la huella digital de objeto, el nodo - h es globalmente único y se deriva del contenido de los objetos de datos que están incluidos en el nodo - h. Cualquier cambio (por ejemplo, adición, eliminación, cambio de metadatos, lectura) da como resultado que se cambie la huella digital de nodo - h de sistema de archivos. Al realizar un seguimiento de los cambios en la correspondencia - i se proporciona una historia completa de toda la actividad de sistema de archivos.

Exclusivo de la invención es un objeto de correspondencia de nodos - i (que también se conoce como correspondencia - i), que convierte un número de nodo - i en una huella digital de objeto. Esto posibilita que el sistema de archivos de espacio de nombres se ocupe de números de nodo - i, lo que es esencial, debido a que muchas actividades de nivel de usuario hacen referencia al número de nodo - i. La puesta en correspondencia de nodos - h de huellas digitales (nombres de objeto) con números de nodo - i proporciona una capa adicional de direccionamiento indirecto (o virtualización) frente a una tabla de nodos - i estáticos tradicional. Mediante el uso de esta tabla de direccionamiento indirecto, un número de nodo - i puede permanecer constante, pero el nombre de objeto asociado (la huella digital) puede cambiar a medida que cambia el archivo que se corresponde con el nodo - i. Debido a que la propia correspondencia - i es un objeto, ese nombre también cambiará a medida que se modifica el sistema de archivos. La huella digital de la correspondencia - i es esencialmente una “instantánea” completa del sistema de archivos. Una vez que se tiene la huella digital de instantánea, se puede continuar trabajando en el sistema de archivos (instantáneas grabables), y recordarlo para un uso futuro (por ejemplo, para una recuperación en casos de desastre). También puede publicarse la huella digital de instantánea en otro sistema, que se encuentra en un almacén de objetos distinto. A pesar de que el otro almacén de objetos puede no hospedar completamente la totalidad de los datos de instantánea (objetos), el mecanismo que se describe sigue siendo completamente consistente y utilizable.

Estos y otros beneficios de la presente invención se describirán más en particular en lo sucesivo con referencia a diversas realizaciones de la invención.

Antes de describir ejemplos específicos del nuevo sistema de archivos, que se implementan tanto en el espacio de núcleo como a continuación en el espacio de usuario, se definirá una descripción más general de los diversos componentes que se utilizan en la presente realización.

Almacén de objetos

Un almacén de objetos, en la presente realización, es una recopilación plana de datos opacos (objetos). Cada objeto es único, y tiene unos recuentos de referencia (el número de veces que este es referenciado por el sistema de archivos de espacio de nombres). El nombre de un objeto es un troceo criptográfico del contenido del objeto, es decir, si se cambia el contenido el nombre ha de cambiar.

Cualquier troceo criptográfico lo bastante fuerte es aceptable para generar nombres de objeto (huellas digitales). A modo de ejemplo, las funciones de troceo del Algoritmo de Troceo Seguro (SHA, *Secure Hash Algorithm*) son un conjunto de funciones de troceo criptográficas diseñadas por la Agencia Nacional de Seguridad (NSA, *National Security Agency*) y publicadas por el NIST como una norma de procesamiento de información federal de EE. UU. SHA-I es la mejor establecida de las funciones de troceo de SHA existentes, y se emplea en varias aplicaciones y protocolos de seguridad ampliamente usados.

En la práctica, los tamaños de objeto son por lo general potencias de 2, y varían de 512 bytes (2^9) hasta 1 MB (2^{20}) o más, a pesar de que no hay restricción arquitectónica alguna sobre el tamaño de un objeto.

Un tamaño de objeto típico es de 2 kB (2^{11} bytes). Para un sistema de archivos de 8 TB (2^{43} bytes), que es de 2^{32} objetos, o aproximadamente 2 billones de objetos. La entrada de cada objeto en el índice es de aproximadamente 32 (2^5) bytes, por lo tanto el índice de objetos, suponiendo que este esté densamente agrupado, es de 2^{37}, o 128 GB, o aproximadamente un 2 % del espacio total del sistema de archivos. Pueden usarse otros tamaños de objeto sin pérdida alguna en cuanto a la susceptibilidad de aplicación o la generalidad.

Los objetos se comprimen y se cifran de forma transparente para el usuario del objeto. Los nombres de objeto se basan en unos datos limpios no comprimidos (y una sal opcional). Lo que se almacena en realidad en el objeto es uno de unos datos (limpios), (limpios comprimidos), (limpios, comprimidos y cifrados) o (limpios y cifrados).

Los objetos por lo general se leen / se escriben solo con datos limpios, y la compresión / cifrado tiene lugar de forma interna al almacén de objetos.

El uso de compendios criptográficos fuertes posibilita que los objetos tengan unos nombres globalmente únicos y consistentes. Dos objetos con el mismo nombre tendrán, para todos los fines prácticos, el mismo contenido.

Espacio de nombres

El sistema de archivos de espacio de nombres, en la presente realización, tiene archivos, una estructura de directorios, enlaces, un superbloque, y así sucesivamente.

El sistema de archivos de espacio de nombres no contiene datos directamente, en su lugar todos los datos se almacenan en objetos. Los objetos son relativamente pequeños, y con frecuencia se necesitan unas estructuras de datos más grandes. La estructura que agrega objetos se denomina nodo - h.

Como una forma práctica, un sistema de archivos que se conecta a un entorno de Unix o de Linux necesita exponer los números de nodo - i. Los nodos - i son unos números que identifican de forma única un archivo.

nodo - h

Un nodo - h, en la presente realización, es una estructura de datos que vincula entre sí un contenido, tal como un archivo. A veces, el contenido puede ser muy grande (de muchos GB), y no cabe de forma contigua en un disco o medio persistente. El contenido se divide, y se almacena como unidades discretas. En el caso de los sistemas de archivos tradicionales, esto serían bloques en disco. En la invención, estos son nombres de objeto. El nodo - h mantiene una lista de la totalidad de los nombres de objeto en una estructura de puesta en correspondencia. Las listas lineales son un ejemplo de una estructura de puesta en correspondencia de este tipo, pero también son posibles unas tablas de direccionamiento indirecto más complicadas.

Hay dos diferencias principales entre un nodo - h y un nodo - i. La primera es que un nodo - h usa nombres de objeto (huellas digitales) que identifican el contenido del objeto, mientras que un nodo - i usa un direccionamiento de bloques físicos o lógicos. La segunda es que un nodo - h tiene un nombre bien definido y globalmente único (el troceo de su contenido). En una realización preferida, que se describe en lo sucesivo, el nombre de nodo - h es un troceo del contenido de objeto y la sal.

Objeto de correspondencia de nodos - i (Correspondencia - i)

Exclusivo de la invención es una correspondencia - i, que convierte un número de nodo - i en una huella digital (nombre) de objeto. Esta huella digital es por lo general un nodo - h, que a su vez se interpreta de diversas formas dependiendo del contexto. Esto posibilita que el resto del sistema de archivos de espacio de nombres se ocupe de números de nodo - i, lo que es esencial, debido a que muchas utilidades de nivel de usuario necesitan ver una construcción de este tipo. En un cierto sentido, esto proporciona una capa adicional de direccionamiento indirecto (o virtualización) frente a una tabla de nodos - i estáticos tradicional.

Mediante el uso de esta tabla de direccionamiento indirecto, un número de nodo - i puede permanecer constante, pero el nombre de objeto asociado (la huella digital) puede cambiar a medida que cambia el archivo que se corresponde con el nodo - i. Debido a que la propia correspondencia - i es un objeto, ese nombre también cambiará a medida que se modifica el sistema de archivos.

En un sistema de archivos tradicional, el directorio raíz se encuentra en un número de nodo - i conocido, y en el caso de la correspondencia - i, que también es el caso.

Si se tiene una huella digital de la correspondencia - i, se tiene esencialmente una “instantánea” completa del sistema de archivos. Forzar el recuento de referencia de cada objeto visible por debajo de esta huella digital bloquea la instantánea, y evita que esta se elimine con independencia de otra actividad de sistema de archivos.

Una vez que se tiene una huella digital de instantánea, se puede continuar trabajando en el sistema de archivos (instantáneas grabables), recordarlo para un uso futuro (quizá con fines de recuperación en casos de desastre). También puede publicarse la huella digital de instantánea en otro sistema, que se encuentra en un almacén de objetos distinto. Si un almacén de objetos no puede resolver una solicitud de lectura de una huella digital particular, hasta el punto de que este está al tanto de otros almacenes de objetos, este puede reenviar la solicitud a estos almacenes. Por lo tanto, la huella digital de la instantánea puede moverse a un sistema cuyo almacén de objetos puede no hospedar completamente la totalidad de los datos (objetos) de la instantánea, pero por medio del mecanismo que acaba de describirse sigue siendo completamente consistente y utilizable.

Superbloque

Un superbloque, en la presente realización, es una estructura de datos que se usa cuando un almacén de objetos reside en medios persistentes. Este reside en una ubicación o ubicaciones conocidas. Este describe en dónde las correspondencias de asignación, la correspondencia - i, la agrupación de objetos, el índice y otras estructuras residen en el medio. Un almacén de objetos siempre tiene un identificador globalmente único (GUID, *globally unique identifier*), que representa esa instancia única de un almacén de objetos.

En el caso donde el almacén de objetos participa en una gran agrupación de objetos, el superbloque también contiene el GUID de la agrupación más grande, y los GUID de la totalidad de los miembros, y la relación de los miembros (despojados, replicados, con codificación de borrado, etc).

Archivo

Una construcción de archivo, en la presente realización, se deriva de un nodo - h. Esta tiene la totalidad de la semántica normal (por ejemplo, de POSIX®) con respecto a archivos, tal como leer, escribir, abrir, cerrar, y así sucesivamente.

Directorio

Un directorio, en la presente realización, es una versión especializada de un nodo - h. Este contiene una correspondencia de pares de (número de nodo - i, nombre de objeto). Una lista lineal, un vector u otras estructuras más complicadas son unas implementaciones a modo de ejemplo. La correspondencia como mínimo ha de ser serializable y deserializable con el fin de persistir como este para un nodo - h. Dependiendo de la estructura de puesta en correspondencia, un acceso aleatorio también es posible.

Seguimiento

A medida que un sistema de archivos se modifica debido a escrituras, eliminaciones y lecturas normales (obsérvese que una lectura cambia los tiempos de acceso), los objetos y los nodos - h que constituyen ese sistema de archivos también cambian. Esto da como resultado una historia de troceos raíz, que a una granularidad muy fina se denomina protección de datos continua (CDP, *continuous data protection*), y con una granularidad más basta, instantáneas. La diferencia se encuentra sólo en la frecuencia con la que se capturan los troceos raíz.

Ha de poder accederse a cada objeto en el sistema a través de por lo menos un troceo raíz.

En la presente realización, a medida que se escribe un nodo - h H, se crea un nuevo nodo - h H', y si tienen lugar más cambios, posiblemente H". Estos cambios puede acumularse, pero llegados a un cierto punto el último cambio se propaga de vuelta hacia arriba hasta la raíz. Esta entrada / salida (10) pendiente posibilita que el sistema de archivos acumule cambios y no se propague hacia arriba hasta la raíz en cada cambio. La frecuencia con la que tiene lugar esto está basada en la política. Hay que abordar en consecuencia los recuentos de referencia para los objetos en la parte intermedia de la lista de cambios H -> H' -> H" de tal modo que no haya referencias pendientes, u objetos que no puedan alcanzarse.

C. Ejemplos (Implementaciones) de la invención

Haciendo referencia a continuación a la figura 1, se muestran diversos componentes de almacenamiento en un núcleo de sistema operativo 101. A pesar de que está dibujado a partir de un entorno de Linux, el diagrama es lo bastante genérico como para que este sea de aplicación a otros sistemas operativos tales como Windows®, Solaris® y otros sistemas operativos de clase Unix.

Se muestra un ejemplo de un sistema de archivos de estilo POSIX® 104, donde POSIX® puede ser uno cualquiera de cualquier número de sistemas de archivos tales como ResierFs, Exts, btrfs y zfs sin pérdida alguna en cuanto a la generalidad. Una capa de sistema de archivos virtual (VFS, *virtual file system*) 103 se usa para extraer por abstracción muchas características comunes de los sistemas de archivos, y proporciona una interfaz consistente 160 al espacio de usuario 100 y otros componentes. El VFS 103 también tiene una interfaz "de borde inferior" bien definida 150 que cualquier sistema de archivos ha de usar (si este espera que la capa del VFS 103 lo reconozca). En la práctica, hay por lo general muchos sistemas de archivos trabajando en paralelo.

Los sistemas de archivos normalmente se sitúan encima de una abstracción de almacenamiento de bloques, que se implementa por unas unidades de accionamiento de bloque 105. El almacenamiento de bloques puede ser en un dispositivo de almacenamiento local de número de unidad lógica LUN, *Logical Unit Number*, 109, o este puede ser en un LUN remoto usando un protocolo de iSCSI. Las unidades de accionamiento de bloque 105 también tienen unas interfaces bien definidas en un sistema operativo.

En la presente realización, el nuevo sistema de archivos trabaja junto a los otros sistemas de archivos en el núcleo. El nuevo sistema de archivos se compone de un sistema de archivos de espacio de nombres 107 que se apila encima de un sistema de archivos de objeto 108 ligero. La interfaz 152 entre los dos componentes puede ser cualquiera de diversas interfaces de objeto de normas de la industria tales como la norma de objetos ANSI T-10.

El sistema de archivos de objeto (almacén de objetos) 108 a su vez se divide de tal modo que se extrae por abstracción una librería de funciones de uso común, la librería de compendio, indexación, compresión, cifrado (DICE, *Digest, Indexing, Compression, Encryption*) 310. La librería 310 puede realizarse completamente en soporte lógico, o aprovechar una diversidad de técnicas de aceleración de soporte físico 113, una de las cuales se ilustra.

El sistema de archivos de objeto 108 crea un contenedor de objetos que puede estar situado encima de un LUN en bruto, una partición en un disco, o un archivo grande. Este también puede hacer referencia a contenedores por medio de una pila de red 106 usando protocolos tales como iSCSI u otros protocolos de bloque de acceso remoto (siendo FCoE otro ejemplo). Un sistema de archivos de red (NFS, *Network File System*) 102 está situado encima de la pila de red 106 (por medio de la interfaz 154) y el NFS está conectado con el VFS 103. La pila de red 106 está conectada con el LUN 109 por medio de la interfaz 160, y con la Nube 110 por medio de la interfaz 159.

Haciendo referencia a la figura 2, el almacén de objetos 108 se descompone adicionalmente. El almacén de objetos 108 contiene unos objetos binarios y opacos, ejemplos de los cuales son P 201, Q 202 y R 203. Los objetos pueden ser de un tamaño variable, a pesar de que en una implementación preferida estos son potencias de 2. Un objeto reside con un cierto desplazamiento en el contenedor, que puede ser un desplazamiento en bytes, o el resto entero de la división de un desplazamiento entre el tamaño de objeto más pequeño (es decir, si el objeto más pequeño es de 512 bytes, entonces el desplazamiento se multiplicaría por 512 para obtener el desplazamiento en bytes).

Cada objeto tiene un nombre (una huella digital), que es un compendio criptográfico (troceo) de la totalidad del contenido del objeto, más una cierta sal específica del sitio. En la figura 2, los nombres de objeto se indican mediante H(P), H(q) y H(r).

Una estructura de índice 204 realiza un seguimiento de nombres de objeto, ubicaciones de objeto y referencias de objeto. La referencia de un objeto se incrementa cada vez que se escribe el objeto. El sistema de archivos de espacio de nombres 107 puede generar lo que este cree que son muchas copias del mismo objeto; el almacén de objetos 108 solo almacena una, pero realiza un seguimiento de cuántas cree en realidad el espacio de nombres que tiene.

El almacén de objetos 108 tiene varias clases de interfaz. La interfaz de lectura, escritura, eliminación 152a hace exactamente eso para los objetos. Una eliminación de objeto en este contexto es en realidad una disminución del recuento de referencia del objeto. El almacenamiento para el objeto en el interior del almacén de objetos se liberará solo cuando el recuento de referencia pase a 0.

Las operaciones de indexación 152b posibilitan una enumeración de objetos por nombre, unos ajustes de recuentos de referencia y una consulta de objetos por nombre.

El almacén de objetos 108 tiene una semántica transaccional (propiedades ACID), y las fronteras de transacción se gestionan a través de las operaciones transaccionales 152c. Esto incluye el inicio, la confirmación y la anulación de una transacción, además de la enumeración de de transacciones pendientes.

Una interfaz de aprovisionamiento 152d posibilita que se creen, se eliminen, se fusionen, se dividan y se agreguen almacenes de objetos.

El índice 204 es una correspondencia, cuya clave primaria es el nombre de objeto. Tal como se analiza en otra parte del presente documento, el índice puede ser muy grande. Hay una entrada de índice para cada objeto en el sistema. Cada entrada contiene:

a) una huella digital del contenido del objeto. Las huellas digitales se generan por un compendio criptográfico sobre el contenido, con una pequeña cantidad de contenido adicional (“sal”) anexado. La sal es común a todos los objetos en el almacén de objetos.

b) un recuento de referencia que indica cuántas veces es referenciado el objeto. El recuento de referencia puede usar aritmética de saturación para ahorrar espacio. Por ejemplo, este puede usar solo 8 bits para realizar un seguimiento de referencias: el recuento de referencia aumentarse y disminuirse, pero si este iguala o supera 255, el recuento “se satura”, y no se permite disminución adicional alguna.

c) un localizador físico. Si el objeto se encuentra en un disco físico, este puede ser un número de bloque lógico LBN. Si el objeto es hospedado por un proveedor de hospedaje (por ejemplo, Amazon S3), entonces este sería una referencia al objeto en la nube.

d) unas banderas para diversos usos. Una bandera indica si el objeto se almacena comprimido o no, otra si se cifra o no. Se encuentran disponibles otras banderas, pero no están asignadas a un uso específico.

La correspondencia de asignación 220 es una correspondencia de bits normal que se usa para bloques asignados en el contenedor de objetos 206.

El contenedor de objetos 206 es una abstracción de almacenamiento persistente aleatoriamente direccionable. Los ejemplos incluyen un LUN en bruto, un archivo, una partición en un disco, o un dispositivo de iSCSI a lo largo de la red de área extensa WAN.

El contenedor de objetos 206 tiene varios componentes 207 - 211 (que no se muestran a escala). Aparte del bloque de descriptores de contenedor 207, que va con un desplazamiento conocido, el orden de los otros componentes no es de una importancia sustancial.

El índice 208 puede tener unas porciones residentes en contenedor, o unas porciones en la memoria 204, o ambas, tal como un árbol-B. La correspondencia de asignación 210 también puede ser parcialmente en disco y en la memoria 220. La migración entre los dos puede lograrse con técnicas de paginación.

A medida que se modifica el almacén de objetos, un registro de transacciones 211 se conserva en un almacenamiento persistente. El registro realiza un seguimiento de toda la actividad de objetos, incluyendo lecturas, escrituras, eliminaciones, ajustes de referencia, y así sucesivamente. El registro se conserva en orden de tiempos, y se anexa de forma periódica al índice principal 208. La actividad de objetos ha de “alcanzar” el registro en primer lugar antes de examinar el índice principal. Cada entrada de registro consiste en un tipo de operación 152a, 152b, 152c, 152d, la huella digital, el recuento de referencia, el ID de transacción o número de época, y la ubicación de agrupación. Una entrada de registro es estructuralmente similar a una entrada de índice, con la adición del ID de transacción.

La denominación de objetos global posibilita que un almacén de objetos mueva objetos de un lugar a otro a la vez que se sigue conservando una denominación y un acceso consistentes. Las razones para mover un objeto incluyen:

a) Mover objetos relacionados uno cerca de otro en un disco físico, por razones de rendimiento.

b) Replicar objetos a lo largo de fronteras de fallo. Esto puede ser a lo largo de dos discos locales separados, un disco local y un disco remoto, o cualquier múltiplo de los mismos. La replicación también puede conferir unos beneficios de rendimiento de lectura. La replicación también puede incluir fraccionar objetos, tal como con códigos de borrado.

c) Operaciones de segundo plano sobre objetos tales como compresión, descompresión, cifrado, descifrado, y así sucesivamente.

d) Mover objetos basándose en la temperatura, es decir, su frecuencia o frecuencia esperada de uso.

La figura 3 ilustra la relación del almacén de objetos 108 con la librería DICE 310. La librería 310 extrae por abstracción unas características comunes del almacén de objetos, tal como los compendios 153a, indexación 153b, compresión 153c y cifrado 153d.

A la vez que se proporciona una interfaz consistente, internamente la librería puede usar una diversidad de técnicas para entregar los servicios. Las técnicas de implementación incluyen solo soporte lógico, asistente de soporte físico parcial (Intel QuickAssist®, por ejemplo), o una implementación de soporte físico personalizada que puede almacenar grandes cantidades de índice, o cualquier combinación de lo anterior.

Si se usa un acelerador de soporte físico 113, ese acelerador puede tener dos clases amplias de servicio: una para unas operaciones con uso intensivo de cómputo 111 (compresión, cifrado, marcado con huella digital), y otra para unas operaciones con uso intensivo de memoria 112 tal como un índice. Una implementación de soporte físico puede tener uno o el otro, o ambos.

La figura 4 ilustra unos componentes clave de una estructura de nodo - h 401 en la presente realización. El nodo - h usa unos identificadores de objeto (huellas digitales) para identificar contenido, en lugar del direccionamiento de bloques físicos / lógicos que usan los nodos - i heredados.

Un nodo - h es una secuencia de contenido, como un archivo, que puede leerse, escribirse, anexarse a, crearse, eliminarse y truncarse de forma aleatoria. Puede accederse al contenido en unas fronteras de bytes arbitrarias, y con unos intervalos arbitrarios. Cómo se interpreta el contenido depende del contexto.

Un nodo - h 401 puede tener una estructura stat 420, por ejemplo, una estructura POSIX® que se usa para los metadatos de archivo. Parte de esa estructura puede incluir la longitud en bytes del archivo, o el nodo - h en este caso. La secuencia de datos se divide en unos objetos discretos, por ejemplo, S 410, T 411 y U 412 en la figura 4. Los nombres de cada objeto se almacenan en una tabla de puesta en correspondencia 402, que registra las huellas digitales de cada uno de S, T y U. No es necesario que los objetos sean de la misma longitud.

La tabla de puesta en correspondencia 402 puede tener diversas representaciones, incluyendo una lista lineal, una estructura en árbol, o una estructura de direccionamiento indirecto, sin pérdida alguna en cuanto a la generalidad. Una tabla de puesta en correspondencia 402 está indexada por un desplazamiento en el contenido (la secuencia S, T, y U) para determinar qué objeto u objetos van a ser referenciados, de una forma similar a aquella donde funcionan las tablas de direccionamiento indirecto de nodos - i de Unix convencionales.

Un nodo - h es, en sí mismo, un objeto, y por lo tanto tiene un nombre único. Cuando cambia uno o más cualesquiera de la estructura stat 420, la tabla de puesta en correspondencia 402, y cualquiera de los objetos referenciados, entonces también cambiará el nombre (la huella digital) del nodo - h.

Puede accederse a un nodo - h de forma aleatoria tanto para lectura como para escritura y anexión. Los nodos - h soportan un espacio disperso, donde los datos que no se han escrito devuelven un valor conocido (por lo general 0).

Cualquier cambio en un nodo - h da como resultado un nuevo nodo - h, debido a que el nombre del nodo - h es una función de su contenido. El nodo - h original puede ser desreferenciado, o conservarse (mediante el aumento del recuento de referencia), dependiendo de la política del sistema de archivos.

Un nodo - h 401 puede tener unas estructuras adicionales, por ejemplo, además de una estructura “stat” de Unix 420 convencional.

Tal como se muestra en la figura 5, un nodo - h 401 es una secuencia aleatoriamente direccionable de contenido, similar a un archivo. Cómo se interpreta ese contenido depende del contexto. En la presente realización de la invención, un nodo - h se especializa adicionalmente en archivos, directorios y correspondencias - i. En la jerga de la programación orientada a objetos, las clases archivo, directorio y correspondencia - i se derivan del nodo - h de clase base.

Un archivo 504 puede ser un contenedor delgado que hace que un nodo - h aparezca como un archivo POSIX® normal que puede abrirse, cerrarse, leerse, escribirse, y así sucesivamente.

Un directorio 505 es otra interpretación de un nodo - h 401. Un directorio 505 es una puesta en correspondencia 501 de números de nodo - i (un número entero) con nombres de archivo (una cadena). La puesta en correspondencia puede adoptar diversas formas, incluyendo pero sin limitarse a, una lista lineal, árboles-B y correspondencias de troceo. Si la correspondencia 501 se encuentra completamente en memoria, es un requisito que la correspondencia pueda serializarse y deserializarse.

Una correspondencia - i (“correspondencia de nodos - i”) 502 traduce números de nodo - i (a partir del directorio 501) a un compendio de objeto (una huella digital). El objeto puede representar un nodo - h (y, por lo tanto, por extensión, un archivo, un directorio u otra correspondencia - i), una estructura tal como un superbloque, u otros datos.

Una correspondencia - i puede tener unas ubicaciones reservadas, tales como el índice 0, el índice 1, y así sucesivamente, para objetos bien conocidos. Los ejemplos incluyen correspondencia - i o correspondencias - i previas, superbloques de sistema de archivos, y así sucesivamente.

La figura 6 ilustra cómo cambian el contenido de los archivos y los metadatos desde un instante inicial T_0 610 hasta el instante T_1 611 a medida que se añade un contenido. La eliminación del contenido sigue una trayectoria similar.

El diagrama muestra tanto componentes del almacén de objetos 108, como componentes del espacio de nombres 107, que están separados por la interfaz 152.

En el instante T_o 610, el directorio $Raíz_o$ $Raíz_o$ 640 tiene dos archivos FOO 641 y BAR 642. El archivo FOO 641 a su vez está compuesto por un contenido que se divide en los objetos P 652 y Q 655. Los nombres de objeto para P 652 y Q 655 se almacenan en la tabla de puesta en correspondencia del FOO 641, que se ha ilustrado previamente (la figura 4). De forma similar, el archivo BAR 642 tiene un contenido Q 655. El directorio raíz 640 también es un objeto, que se indica mediante $Raíz_o$ 653. De forma similar, los archivos (nodos - h) FOO 641 y BAR 642 están representados en los objetos 651 y 654 respectivamente. La Correspondencia - i_0 inicial 502a también está representado en una Correspondencia - i_0 de objeto 650, como lo es el directorio raíz $Raíz_0$ 640 que tiene un objeto $raíz_0$ 653.

Debido a que el objeto Q 655 es común a los archivos tanto FOO 641 como BAR 642, este tiene un recuento de referencia de 2, mientras que el objeto P 652 solo tiene un recuento de referencia de 1 en el instante T_0 610.

El directorio raíz 640 contiene dos entradas, una para cada uno de FOO y BAR. La entrada de FOO tiene un índice de nodos - i de 4, y el índice de nodos - i del BAR es 9.

La correspondencia - i_0 502a es un nodo - h, y se almacena en ese sentido como un objeto 650. Para evitar complicar el dibujo, a pesar de que la correspondencia - i es un nodo - h, y un nodo - h puede ponerse en correspondencia con muchos objetos, este se muestra en el presente caso como un objeto.

Por convención, el compendio del directorio raíz siempre se almacena en el índice de correspondencia - i 2. El compendio de una correspondencia - i posibilita un acceso completo a un sistema de archivos. Mediante la lectura del objeto que está asociado con la correspondencia - i, se obtiene el directorio raíz, y a partir de allí cualquier

directorio y / o archivos subsiguientes. Además, el compendio de una correspondencia - i define con precisión y de manera no ambigua el contenido de la totalidad del sistema de archivos de aguas abajo.

Inmutabilidad: si, por ejemplo, el objeto Q cambia, entonces el nombre cambia (el nombre de un objeto es una función de su contenido). Cualesquiera tablas de puesta en correspondencia que apunten al Q modificado ahora no lo hacen, y por lo tanto el Q modificado no es “visible”. Unos argumentos similares son de aplicación a cualquier objeto al que pueda hacerse referencia mediante el compendio de una correspondencia - i.

En el instante T_1 611, el archivo BAR 642 tiene un contenido S 658 anexado al mismo, de tal modo que se crea un nuevo archivo BAR 644. Un nuevo archivo BAR ha de crearse de tal modo que los compendios y los nombres de objeto son consistentes. A medida que se añade un nuevo contenido S 658, cada cosa que haga referencia a este también se actualiza y se crea una versión nueva. Esto es de aplicación a una versión más nueva de BAR 644, el directorio raíz 643, y lo más importante, a una nueva tabla de correspondencias - i 502b. Los recuentos de referencia de objeto 614 en el instante To 610 se ajustan a medida que se añade / se retira un contenido, de tal modo que en el instante T_1, T_1 los recuentos de referencia de objeto 615 representan un contenido que es exclusivo de To, exclusivo de T_1 y un contenido que se encuentra en común.

En el instante T_1 611, hay esencialmente dos sistemas de archivos que tienen una gran cantidad de contenido común. Los dos sistemas de archivos se especifican completamente por los compendios de sus correspondencias - i, la correspondencia - i_0 502a y la correspondencia - i_1 502b respectivas. Por ejemplo, en el instante To 610 el objeto Q 655 puede ser referenciado a través de las trayectorias (640a, 641 b), (640b, 642a), (643a, 641 b) y (643b, 644a).

A medida que el contenido de un archivo se modifica (se añade, se borra, se modifica), la tabla de puesta en correspondencia del archivo también se cambia. A su vez el objeto que contiene la puesta en correspondencia de archivos, el nodo - h, también cambia. Por diversas razones (rendimiento, interfaces de gestión), puede que no sea apropiado propagar cada cambio hasta arriba del árbol hasta el directorio raíz y a la correspondencia - i. No obstante, si se hace en cada transacción de E / S, el sistema implementa de forma implícita una CDP, donde cada compendio de la correspondencia - i representa una transacción de E / S particular. Si se hace de forma periódica (por ejemplo, cada hora o así), a demanda, o al ocurrir sucesos particulares (cierre de archivo), entonces el comportamiento es similar a las instantáneas de sistema de archivos.

Debido a que los objetos tienen unos recuentos de referencia, hasta el punto de hay objetos idénticos, la desduplicación es nativa al sistema. A medida que un sistema de archivos cambia como resultado de modificaciones, en su mayor parte, solo los cambios darán como resultado que se añada un nuevo contenido a la agrupación de almacenamiento.

En la figura 7, el objeto 701 es una secuencia de datos opacos y binarios y tiene un nombre de objeto 703. Los tamaños de objeto son arbitrarios, a pesar de que en una implementación preferida estos pueden ser de un tamaño de una potencia de 2 para hacer la asignación y la gestión de almacenamientos persistentes más fácil.

Para el usuario del objeto, el contenido siempre se lee, se escribe y se accede al mismo como el contenido de objeto limpio 710. El almacén de objetos almacena internamente el objeto en una forma que puede incluir una compresión 711 y / o un cifrado 712 opcionales. Por lo tanto, lo que puede parecerle al usuario como un objeto de 2048 bytes se almacena internamente como 512 bytes de datos (suponiendo una relación de compresión de 4 : 1), que se cifran adicionalmente. Un almacén de objetos es un dominio de cifrado, lo que quiere decir que todos los objetos se tratan de forma similar con respecto al cifrado. Esto es distinto de cualquier cifrado que puedan usar los llamantes del objeto.

En la figura 8, se ilustra una realización alternativa de un sistema de archivos que se implementa en el espacio de usuario 100. Usando el marco FUSE de código abierto, el sistema de archivos de espacio de nombres 107 se enlaza contra la librería FUSE de modo de usuario 802. El sistema de archivos de espacio de nombres tiene la misma interfaz privada 152 con el almacén de objetos 108. El almacén de objetos 108 también tiene la misma interfaz 153 con la librería DICE 310. La librería DICE puede usar opcionalmente el asistente de soporte físico 113. El sistema de archivos vertical 103 reside en el núcleo 101, al igual que lo hace el asistente de soporte físico 113 y el módulo de FUSE 801 (conectado tanto con el VFS 103 como con la librería FUSE 802).

D. Realizaciones alternativas

Una forma novedosa de construir un sistema de archivos que integra una combinación de características a una fracción del coste de los sistemas anteriores se ha descrito en lo que antecede. Diversas modificaciones serían evidentes para el experto en la construcción de realizaciones alternativas.

El nuevo sistema de archivos puede realizarse en una forma de soporte lógico puro, ejecutándose en un ordenador como cualquier otro sistema de archivos. Además, la propia organización del sistema de archivos integrado se presta a unas técnicas de aceleración de soporte físico únicas que no son posibles con los sistemas de archivos

heredados. La aceleración de soporte físico posibilita más rendimiento para un coste dado, o un coste total inferior de titularidad para un nivel de rendimiento dado.

En la realización anterior, el sistema de archivos proporciona un conjunto de características integradas. El sistema de archivos se implementa como una pila que incluye dos sistemas de archivos distintos, un sistema de archivos de objeto y un sistema de archivos de espacio de nombres. La pila es completamente conforme con POSIX®, y puede usarse siempre que se exija un sistema de archivos conforme con POSIX®, tal como el segundo sistema de archivos extendido (EXT2), el tercer sistema de archivos extendido (EXT3), ReiserFs, y así sucesivamente.

La porción inferior de la pila es un sistema de archivos de objeto. El sistema de archivos basado en objeto se usa para hospedar los datos en forma de objetos. Un objeto es una secuencia de datos opacos y binarios. El objeto puede ser datos en bruto, o metadatos (por ejemplo, un registro de la creación de y cualesquiera cambios en los datos en bruto). El tamaño de objeto puede variar, pero por lo general está limitado a un intervalo de unos pocos kilobytes (kB); no obstante esto no se requiere para un funcionamiento correcto de la invención. El nombre (al que también se hace referencia en el presente documento como huella digital) del objeto se deriva del contenido del objeto usando por ejemplo un troceo criptográfico fuerte. Esto posibilita que el nombre de objeto sea globalmente único e identificable, es decir, una huella digital del contenido. El sistema de archivos de objeto está principalmente orientado a machines.

Dos huellas digitales que son iguales representarán, para todos los fines prácticos, el mismo contenido, con independencia de en dónde se calcularon las huellas digitales. A la inversa, dos huellas digitales que son diferentes representan un contenido diferente. Debido a que las huellas digitales son significativamente más pequeñas que los objetos (por ejemplo, un factor de 100 X, 1000 X o más), manipular las huellas digitales es con frecuencia más rápido y más fácil que manipular el contenido subyacente.

El sistema de archivos de objeto que se ha descrito en la realización anterior es ligero y plano, distinto de los sistemas de archivos de objeto pesados tal como se describe en la especificación ANSI T-10, o los sistemas de archivos direccionables por contenido tales como los EMC Centera®, comercialmente disponibles, o un producto de Hitachi (adquisición por medio de Archivas). Objetos, tal como se usa en el presente caso, no debería confundirse con objetos tal como se usa en lenguajes de programación tales como C++ y Java.

Los sistemas de archivos de objeto tienen un “índice” que realiza un seguimiento de la totalidad de los objetos. La construcción y la gestión de un índice de este tipo pueden ser un desafía importante para los sistemas de archivos de objeto, donde puede haber muchos millones, o incluso billones de entradas en el índice.

De acuerdo con la realización que se describe, se proporciona en la parte superior de una pila de almacenamiento un sistema de archivos de espacio de nombres que tiene archivos, directorios y así sucesivamente. Una diferencia con respecto a lo conocido (por ejemplo, sistemas de archivos POSIX®) no obstante es que en lugar de usar un direccionamiento de número de bloque lógico (LBN) para acceder a un contenido, se usan huellas digitales de objeto. Además, todas las estructuras de datos internas del sistema de archivos de espacio de nombres son ellas mismas objetos. Por lo tanto, la totalidad de la pila de almacenamiento (el espacio de nombres y la capa de objetos) se “enlaza” entre sí mediante referencias de objeto, y tener la huella digital del objeto que representa la raíz posibilita que se defina completamente y de manera no ambigua la totalidad de la estructura de archivos.

Cualquier cambio (adiciones, eliminaciones, cambio de metadatos, lecturas) da como resultado que se cambie la firma del sistema de archivos. Al realizar un seguimiento de la firma raíz, por lo tanto se puede obtener una historia completa de toda la actividad de sistema de archivos.

De acuerdo con la realización que se divulga de la invención, la división de trabajo en dos componentes separados (el espacio de nombres 107 y el almacén de objetos 108) y cómo interaccionan estos, se realiza de una forma tal que la desduplicación, las instantáneas, las instantáneas grabables, la protección de datos continua (CDP, *continuous data protection*), la eficiencia de red de área extensa, el control de versiones, la comprobación de integridad de sistema de archivos y la inmutabilidad disminuye de forma natural, a la vez que se sigue conservando la semántica de POSIX®.

De acuerdo con la realización que se divulga, la organización del sistema de archivos posibilita la aplicación de asistente de soporte físico. El asistente de soporte físico puede adoptar dos formas. Una forma es para el cómputo de la aceleración, tal como compresión, cifrado y compendios criptográficos. La segunda forma es para la construcción y el mantenimiento para un gran índice que a su vez se usa para construir un almacén de objetos práctico.

Se gastan unos recursos de CPU significativos en la aplicación de troceo criptográfico, compresión y cifrado. Unos relojes de CPU más rápidos y más núcleos de CPU mitigan esto hasta cierto punto, pero debido a que aumentan los requisitos de rendimiento, es deseable descargar parte o la totalidad de estas funciones en un soporte físico dedicado (aceleración). Hay varios conjuntos de chips comerciales (por ejemplo, Hifn, Cavium) que pueden lograr esto.

El índice de almacén de objetos puede ser grande, y puede superar con rapidez los límites de memoria prácticos. Un índice de objetos global (es decir, un índice para la totalidad del almacenamiento) que se lee y se escribe de forma aleatoria (la clave primaria para un índice de este tipo es un troceo criptográfico, que tiene una distribución aleatoria), puede hacer los algoritmos de paginación y de almacenamiento en memoria caché no efectivos. La colocación de un índice de este tipo en un almacenamiento no volátil más rápido, tal como un disco de estado sólido (SSD) por lo tanto proporcionaría unos beneficios de rendimiento.

Los SSD se construyen de tal modo que las tasas de lectura son significativamente más altas que las tasas de escritura (es decir, Seagate xxx puede entregar 35.000 IOPS / lectura y 3000 IOPS / escritura). Si el acceso al índice se divide de manera uniforme entre lecturas y escrituras, entonces no se obtienen muchos de los beneficios de un SSD.

Una solución de indexación de construcción personalizada, hecha de FLASH y una FPGA puede aumentar el ancho de banda de indexación incluso más. 310

El asistente de soporte físico puede ser gestionado por la librería DICE tal como se ha descrito previamente.

Realizaciones de la invención pueden implementarse en circuitería electrónica digital, o en soporte físico, soporte lógico inalterable, soporte lógico de ordenador, o en combinaciones de los mismos. Realizaciones de la invención pueden implementarse como un producto de programa informático, es decir, un programa informático que está incorporado de forma tangible en un soporte de información, por ejemplo, en un dispositivo de almacenamiento legible por máquina, para la ejecución por, o para controlar la operación de, un aparato de procesamiento de datos, por ejemplo, un procesador programable, un ordenador, o múltiples ordenadores. Un programa informático puede escribirse en cualquier forma de lenguaje de programación, incluyendo lenguajes compilados o interpretados, y este puede desplegarse en cualquier forma, incluyendo como un programa independiente o como un módulo, un componente, subrutina, u otra unidad adecuada para su uso en un entorno informático. Un programa informático puede desplegarse para ejecutarse en un ordenador o en múltiples ordenadores en un sitio o distribuirse a lo largo de múltiples sitios e interconectarse mediante una red de comunicaciones.

Las etapas de método de realizaciones de la invención pueden realizarse por uno o más procesadores programables que ejecutan un programa informático para realizar funciones de la invención al operar sobre unos datos de entrada y generar una salida. Las etapas de método también pueden realizarse por, y el aparato de la invención puede implementarse como, una circuitería lógica de propósito especial, por ejemplo, una FPGA (*field programmable gate array*, matriz de puertas programable en campo) o un ASIC (*application specific integrated circuit*, circuito integrado de aplicación específica).

Los procesadores adecuados para la ejecución de un programa informático incluyen, a modo de ejemplo, microprocesadores de propósito tanto general como especial, y uno o más procesadores cualesquiera de cualquier tipo de ordenador digital. En general, un procesador recibirá instrucciones y datos a partir de una memoria de solo lectura o una memoria de acceso aleatorio o ambas. Los elementos esenciales de un ordenador son un procesador para ejecutar instrucciones y uno o más dispositivos de memoria para almacenar instrucciones y datos. En general, un ordenador también incluirá, o estará acoplado de forma operativa para recibir datos a partir de o transferir datos a, o ambos, uno o más dispositivos de almacenamiento masivo para almacenar datos, por ejemplo, discos magnéticos, discos magneto ópticos, o discos ópticos. Los soportes de información adecuados para incorporar datos e instrucciones de programa informático incluyen todas las formas de memoria no volátil, incluyendo a modo de ejemplo, dispositivos de memoria de semiconductores, por ejemplo, dispositivos de memoria EPROM, EEPROM y flash; discos magnéticos, por ejemplo, discos duros internos o discos extraíbles; discos magneto ópticos; y discos CD ROM y DVD-ROM. El procesador y la memoria pueden estar complementados por, o incorporados en una circuitería lógica de propósito especial.

E. Nodo - h (contenido más sal)

En una realización, el nombre de nodo - h es un troceo de su contenido más sal. La sal es un valor pequeño, del orden de 8 a 32 bytes, que se antepone o se anexa de forma interna y automática a cada objeto antes de que se calcule la firma. Esta no se almacena cuando se escribe el objeto.

Por ejemplo, un usuario teclea una contraseña, a partir de la cual se genera la sal usando cualquiera de una diversidad de técnicas convencionales que se usan para la generación de claves criptográficas. Un usuario protegería esta contraseña, al igual que cualquier otra contraseña. Incluso si se obtiene la sal, no es computacionalmente posible generar la contraseña original.

La sal es principalmente un mecanismo de defensa frente a datos con comportamiento erróneo, en este ejemplo donde:

- cada objeto se somete a seguimiento usando su huella digital;
- las huellas digitales se almacenan en un índice;

- la distribución de huellas digitales es plana, es decir, todas las huellas digitales tienen igual probabilidad de tener lugar;
- los algoritmos de indexación se basan en una distribución uniforme de claves (huellas digitales).

Si una entidad malintencionada sabe que el sistema de archivos usa un algoritmo de huella digital específico, póngase por caso SHA-1, la entidad puede generar con facilidad un contenido que tiene unas huellas digitales que entran dentro de un intervalo muy angosto. Para hacer esto, la entidad sigue generando un contenido aleatorio, lo marca con huella digital, y guarda solo el contenido que entra dentro del angosto intervalo especificado. Eso daría lugar a que los algoritmos de indexación tuvieran un rendimiento muy pobre.

No obstante, la naturaleza de los troceos criptográficos es de tal modo que si se cambia solo 1 bit del contenido de un objeto, cambiará aproximadamente un 50 % de los bits de la huella digital. Ese 50 % también se aleatoriza a medida que se cambian diferentes bits del contenido original.

La adición de la sal (es decir, un cambio relativamente pequeño) por lo tanto aleatoriza las huellas digitales, lo que hace muy difícil "cazar" los algoritmos de indexación.

G. Indexación escalonable (realización)

El método y aparato de la invención pueden implementarse con los siguientes algoritmos de indexación y tecnología de memoria que se describen en el documento de EE. UU. pendiente junto con la presente y del mismo propietario que la presente con Nº de serie 12/823.452 con título "*Scalable Indexing*", de los mismos inventores P. Bowden y A. J. Beaverson, presentado en la misma fecha (25 de junio de 2010) que la presente solicitud y que reivindica la prioridad para el documento provisional de EE. UU. con Nº 61/269.633 presentado el 26 de junio de 2009. En el presente documento se reivindica la prioridad para ambas solicitudes y las divulgaciones completas de las cuales se incorporan por la presente por referencia en su totalidad.

Ha de entenderse que las descripciones anteriores y siguientes tienen por objeto ilustrar y no limitar el alcance de la invención.

1) Sumario

De acuerdo con una realización de la invención, se proporciona un método de acceso a un índice que está almacenado en una memoria de acceso no uniforme por un proceso de indexación de acceso uniforme, comprendiendo el método:

mantener una tabla de traducción para poner en correspondencia un identificador de sector de almacenamiento lógico que se genera por el proceso de indexación con una ubicación de sector de almacenamiento físico de la memoria para acceder a cada entrada de datos de registro en el índice;
recopilar en una memoria caché una pluralidad de las entradas de datos de registro, que van a escribirse en el índice, antes de una escritura secuencial subsiguiente de la recopilación de entradas en por lo menos una ubicación de sector de almacenamiento físico de la memoria.

En una realización, el método incluye:

escribir la recopilación de entradas de datos de registro a partir de la memoria caché en una ubicación de sector de almacenamiento de la memoria como una escritura secuencial;
actualizar la tabla de traducción con la ubicación de sector de almacenamiento para las entradas de datos de registro de la recopilación.

En una realización, el método incluye:

leer una o más entradas de datos de registro secuenciales a partir de la memoria a la memoria caché;
designar como libres las ubicaciones físicas en memoria a partir de las cuales se leyeron las una o más entradas.

En una realización, el método incluye:

volver una pluralidad de ubicaciones de sector de almacenamiento físico secuenciales en la memoria como un bloque libre mediante la lectura de cualesquiera entradas válidas en el bloque a la memoria caché y
designar como libres las ubicaciones físicas en memoria a partir de las cuales se leyeron tales entradas.

En una realización:

el proceso de indexación genera unas solicitudes de acceso aleatorio para el índice basándose en unas claves de índice distribuidas de manera uniforme y únicas.

En una realización:

las claves comprenden unos compendios de troceo criptográfico.

En una realización:

el proceso de indexación comprende un proceso de aplicación de troceo de desplazamiento.

En una realización:

la aplicación de troceo de desplazamiento comprende un proceso de aplicación de troceo de cuco.

En una realización:

la memoria comprende uno o más de dispositivos de memoria flash, de cambio de fase y de disco de estado sólido.

En una realización:

la memoria está limitada por uno o más de tiempo de acceso de escritura aleatoria, tiempo de acceso de lectura aleatoria - modificación - escritura, escritura secuencial, restricciones de alineación, tiempo de borrado, fronteras de bloque de borrado y desgaste.

En una realización:

un tamaño del sector de almacenamiento físico comprende un tamaño de escritura mínimo de la memoria.

En una realización:

el tamaño del sector de almacenamiento físico comprende una página o una página parcial.

En una realización:

la memoria tiene un bloque de borrado que comprende una pluralidad de páginas.

En una realización el método incluye:

mantener una tabla de sectores de almacenamiento válidos para realizar un seguimiento de qué ubicaciones de sector de almacenamiento en la memoria son válidas.

En una realización:

un sector de almacenamiento en memoria comprende un conjunto de una o más entradas de datos de registro y un auto-índice en la tabla de traducción de sectores de almacenamiento.

En una realización:

las entradas de datos de registro en el sector de almacenamiento no están ordenadas.

En una realización el método incluye:

designar como de solo lectura en una memoria caché las entradas de datos de registro que se escriben de forma secuencial en la memoria.

En una realización:

la tabla de traducción de sectores de almacenamiento se almacena en una memoria persistente.

En una realización, el método incluye:

realizar un seguimiento del número de sectores de almacenamiento libres en un bloque de borrado e implementar un proceso para generar un bloque de borrado libre cuando se cumple un umbral de sectores de almacenamiento libres.

En una realización:

el proceso de indexación realiza operaciones de indexación basándose en solicitudes que registra qué ha de insertarse, eliminarse, consultarse y / o modificarse.

En una realización:

el proceso de indexación presenta unas operaciones con sectores de almacenamiento lógicos para leer de y escribir en unos sectores de almacenamiento físicos que almacenan los registros del índice.

En una realización:

las operaciones con sectores de almacenamiento físicos incluyen unas lecturas aleatorias y unas escrituras secuenciales.

En una realización:

las operaciones con sectores de almacenamiento físicos incluyen además unas órdenes de recorte.

En una realización:

la memoria comprende una capa de dispositivo físico caracterizada por un acceso de lectura y de escritura no uniforme e inmutabilidad con respecto al tamaño, la alineación y la temporización.

En una realización:

la entrada de datos de registro comprende campos para una clave, un recuento de referencia y una dirección de bloque físico.

En una realización:

la clave comprende un compendio de troceo criptográfico de datos;
el campo de dirección de bloque físico contiene un puntero a la dirección de bloque físico de los datos almacenados en un dispositivo de almacenamiento.

En una realización:

las ubicaciones de sector de almacenamiento lógico se generan por una pluralidad de funciones de troceo.

En una realización:

la memoria comprende un dispositivo de memoria flash que incluye una pluralidad de bloques de borrado, cada bloque de borrado comprende una pluralidad de páginas, y cada página comprende una pluralidad de sectores de almacenamiento.

De acuerdo con otra realización de la invención, se proporciona

un producto de programa informático que comprende unos medios de código de programa que, al ejecutarse por un procesador, realiza las etapas del método anterior.

De acuerdo con otra realización de la invención, se proporciona

un medio legible por ordenador que contiene unas instrucciones de programa ejecutables para un método de acceso a un índice que está almacenado en una memoria de acceso no uniforme por un proceso de indexación de acceso uniforme, comprendiendo el método:

mantener una tabla de traducción para poner en correspondencia un identificador de sector de almacenamiento lógico que se genera por el proceso de indexación con una ubicación de sector de almacenamiento físico de la memoria para acceder a cada entrada de datos de registro en el índice;
recopilar en una memoria caché una pluralidad de las entradas de datos de registro, que van a escribirse en el índice, antes de una escritura secuencial subsiguiente de la recopilación de entradas en por lo menos una ubicación de sector de almacenamiento físico de la memoria.

De acuerdo con otra realización de la invención, se proporciona un sistema que comprende:

un procesador físico y dispositivos de memoria que incluyen un medio legible por ordenador que contiene unas instrucciones de programa ejecutables para un método de acceso a un índice que está almacenado en una memoria de acceso no uniforme por un proceso de indexación de acceso uniforme, comprendiendo el método:

mantener una tabla de traducción para poner en correspondencia un identificador de sector de almacenamiento lógico que se genera por el proceso de indexación con una ubicación de sector de almacenamiento físico de la memoria para acceder a cada entrada de datos de registro en el índice;
recopilar en una memoria caché una pluralidad de las entradas de datos de registro, que van a escribirse en el índice, antes de una escritura secuencial subsiguiente de la recopilación de entradas en por lo menos una ubicación de sector de almacenamiento físico de la memoria.

En una realización:

la memoria que almacena el índice comprende una capa de dispositivo físico caracterizada por un acceso de lectura y de escritura no uniforme e inmutabilidad con respecto al tamaño, la alineación y la temporización.

En una realización:

la memoria que almacena el índice comprende uno o más de dispositivos de memoria flash, de cambio de fase y de disco de estado sólido.

En una realización:

la memoria que almacena el índice comprende un dispositivo de memoria flash que incluye una pluralidad de bloques de borrado, cada bloque de borrado comprende una pluralidad de páginas, y cada página comprende una pluralidad de sectores de almacenamiento.

De acuerdo con otra realización de la invención, se proporciona

un método de acceso a un índice que está almacenado en una memoria de acceso no uniforme por un proceso de indexación de acceso uniforme, comprendiendo el método:

proporcionar a una tabla de traducción, que pone en correspondencia un identificador de sector de almacenamiento lógico con una ubicación de sector de almacenamiento físico de la memoria para cada entrada de datos de registro en el índice, unos identificadores de sector de almacenamiento lógico que se generan por el proceso de indexación;
acceder a unas ubicaciones de sector de almacenamiento físico puestas en correspondencia con los identificadores de sector de almacenamiento lógico;
recopilar en una memoria caché unas entradas de datos de registro que van a escribirse en el índice;
escribir subsiguientemente de forma secuencial una recopilación de las entradas de datos de registro a partir de la memoria caché en el índice en por lo menos una nueva ubicación de sector de almacenamiento físico de la memoria; y
actualizar la tabla de traducción para asociar la por lo menos una nueva ubicación de sector de almacenamiento físico con un identificador de sector de almacenamiento lógico.

De acuerdo con otra realización de la invención, se proporciona un sistema informático que comprende:

una memoria de acceso no uniforme donde se almacena un índice que comprende unas entradas de datos de registro en unas ubicaciones de sector de almacenamiento físico de la memoria;
una tabla de traducción para poner en correspondencia un identificador de sector de almacenamiento lógico que se genera por un proceso de indexación de acceso uniforme con una ubicación de sector de almacenamiento físico de la memoria para cada una de las entradas de datos de registro;
una memoria caché para unas entradas de datos de registro recopiladas que van a escribirse en un índice;
unos medios para acceder a unas ubicaciones de sector de almacenamiento físico de la memoria puestas en correspondencia con unos identificadores de sector de almacenamiento lógico que son suministrados a la tabla de traducción por el proceso de indexación;
unos medios para escribir de forma secuencial una recopilación de las entradas de datos de registro a partir de la memoria caché en el índice en por lo menos una ubicación de sector de almacenamiento físico de la memoria; y
unos medios para actualizar la tabla de traducción para asociar la por lo menos una ubicación de sector de almacenamiento físico con un identificador de sector de almacenamiento lógico.

2) Dibujos

La invención de indexación se entenderá más completamente por referencia a la descripción detallada, junto con las siguientes figuras:

la figura 9 es un diagrama de bloques esquemático que ilustra diversas operaciones de indexación que se realizan de acuerdo con una realización de la presente invención;

las figuras 10A a 10D ilustran diversas realizaciones de estructuras de datos que pueden usarse en la presente invención;

la figura 11 es un diagrama de bloques esquemático que ilustra una operación de consulta de acuerdo con una realización de la invención;

la figura 12 es un diagrama de bloques esquemático que ilustra una operación de inserción de acuerdo con una realización de la invención;

la figura 13 es un diagrama de bloques esquemático de una operación de eliminación de acuerdo con una realización de la invención;

la figura 14 es un diagrama de bloques esquemático de una operación de actualización de acuerdo con una realización de la invención;

las figuras 15A y 15B son unos diagramas de bloques esquemáticos que ilustran un proceso de lectura aleatoria para generar unos bloques de borrado libres de acuerdo con una realización de la invención;

las figuras 16A y 16B son unos diagramas de bloques esquemáticos que ilustran otro método de generación de unos bloques de borrado libres de acuerdo con un proceso de depuración;

la figura 17 es un diagrama de bloques esquemático que ilustra una vista o pila de seis capas para ilustrar una implementación de la presente invención;

la figura 18 es un diagrama esquemático de una entrada de registro tal como se usa en una realización de la invención;

las figuras 19A - 19E ilustran de forma esquemática una implementación de aplicación de troceo de cuco de acuerdo con una realización de la invención;

la figura 20 es una ilustración esquemática de múltiples sectores de almacenamiento, conteniendo cada sector de almacenamiento múltiples registros de acuerdo con una realización de la invención;

la figura 21 es un diagrama esquemático de los contenidos de un sector de almacenamiento de acuerdo con una realización de la invención;

la figura 22 es un diagrama de bloques esquemático que ilustra un ejemplo de un chip de flash físico que tiene múltiples pastillas, bloques de borrado, páginas y sectores de almacenamiento de acuerdo con una realización de la invención; y

las figuras 23A - 23B ilustran determinados componentes de una capa de gestión de dispositivo de acuerdo con una realización de la invención.

3. Visión de conjunto

De acuerdo con una o más realizaciones de la invención, se usan unos algoritmos y tecnología de memoria especializados para construir unos índices que tienen de forma simultánea grandes cantidades de registros y requisitos de transacción. Una realización utiliza un algoritmo de indexación de aplicación de troceo de desplazamiento, por ejemplo una aplicación de troceo de cuco. La invención posibilita el uso de tecnologías de memoria de acceso no uniforme tales como dispositivos de memoria flash, de cambio de fase y de disco de estado sólido (SSD, *solid state disk*).

En diversas realizaciones de la invención, se proporcionan nuevas estructuras de datos y métodos para asegurar que un algoritmo de indexación se comporta de una forma que sea natural (eficiente) para el algoritmo, mientras que el dispositivo de memoria ve unos patrones de E / S (entrada / salida) que son eficientes para el dispositivo de memoria.

Se crea una estructura de datos, una tabla de direccionamiento indirecto, que pone en correspondencia unos sectores de almacenamiento lógicos tal como son vistos por el algoritmo de indexación con unos sectores de almacenamiento físicos en el dispositivo de memoria. Esta puesta en correspondencia es de tal modo que se potencia el rendimiento de escritura para los dispositivos de memoria de acceso no uniforme.

Otra estructura de datos, una memoria caché asociativa, se usa para recopilar sectores de almacenamiento y escribirlos de forma secuencial en el dispositivo de memoria, como parte de las políticas de expulsión y de reescritura de la memoria caché.

Se usan métodos para poblar la memoria caché con sectores de almacenamiento (de registros) que son requeridos por el algoritmo de indexación. Pueden leerse unos sectores de almacenamiento adicionales del dispositivo de memoria a una memoria caché durante una lectura a demanda, o por un proceso de depuración.

El uso de la memoria caché, junto con la tabla de direccionamiento indirecto, permite grandes escrituras secuenciales en el dispositivo de memoria.

A pesar de que la tecnología de flash tiene la aptitud fundamental de lograr la capacidad y las tasas de E / S necesarias para el problema de indexación, las características de acceso de flash son no uniformes. Esta no uniformidad es lo bastante significativa como para que los algoritmos de indexación normales funcionen pobremente, si es que funcionan en absoluto, con un dispositivo de memoria flash.

La memoria flash de acceso no uniforme que se usa en la presente invención es una memoria de solo lectura programable eléctricamente borrable (EEPROM, *electrically-erasable programmable read-only memory*) que ha de leerse, escribirse en y borrarse en unos tamaños de bloque grandes de cientos a miles de bits, es decir, no acceso aleatorio a nivel de bytes. A nivel físico, flash es una forma de memoria no volátil que almacena información en una disposición de células de memoria fabricadas a partir de transistores de puerta flotante. Hay dos tipos de dispositivos de memoria flash, flash de NAND y flash de NOR. Flash de NAND proporciona una densidad más alta y una capacidad más grande con un coste más bajo, con unas velocidades más rápidas de borrado, de escritura secuencial y de lectura secuencial, que flash de NOR. Tal como se usa en la presente solicitud y en la presente invención, la memoria “flash” tiene por objeto cubrir la memoria flash de NAND y no la memoria NOR. NAND incluye tanto los dispositivos de célula de nivel único (SLC, *single-level cell*), donde cada célula almacena solo un bit de información, como los más nuevos dispositivos de célula de múltiples niveles (MLC, *multi-level cell*), que pueden almacenar más de un bit por célula. A pesar de que flash de NAND proporciona unos tiempos de acceso rápidos, esta no es tan rápida como la memoria DRAM volátil que se usa como memoria principal en los PC. Un dispositivo de memoria flash puede o puede no incluir un sistema de archivos de flash. Los sistemas de archivos de flash por lo general se usan con memorias flash embebidas que no tienen un controlador incorporado para realizar una nivelación de desgaste y una corrección de errores.

Un chip de flash de NAND típico puede almacenar varios GB de contenido. A diferencia de la memoria acoplada a un ordenador, a la memoria en el chip de flash ha de accederse en determinados tamaños y en determinadas fronteras. Además, una vez que se ha escrito una sección de memoria, una operación de borrado ha de realizarse antes de que pueda escribirse de nuevo en esas ubicaciones de memoria. Así mismo, las ubicaciones se desgastan, por lo tanto asegurar que todas las ubicaciones obtengan un número similar de escrituras complica adicionalmente el uso. Los tiempos de lectura, los tiempos de escritura y los tiempos de borrado pueden variar de forma significativa (de microsegundos a milisegundos). Por lo tanto la temporización, la nivelación de desgaste y las restricciones de alineación hacen el uso práctico de flash difícil en el mejor de los casos.

Un dispositivo de memoria flash puede contener una o más pastillas (obleas de silicio). Puede accederse a cada pastilla, en su mayor parte, de forma independiente.

Una pastilla se compone de miles de bloques de borrado. Un bloque de borrado es por lo general de un tamaño de 128 - 512 kB. Cuando es necesario eliminar datos, estos han de eliminarse en las fronteras de los bloques de borrado.

Otra limitación de flash de NAND es que los datos solo pueden escribirse de forma secuencial. Además, el tiempo de establecimiento para una escritura es largo, aproximadamente 10 X el de una lectura.

Los datos se leen con una granularidad de páginas. Una página puede variar de 1 kB a 4 kB dependiendo del chip de flash particular. Con cada página están asociados unos pocos bytes que pueden usarse para una suma de comprobación de código de corrección de errores (ECC, *error correcting code*).

Los datos se escriben con una granularidad de páginas. Una vez que se ha escrito, la página puede no escribirse de nuevo hasta que se borra su bloque de borrado (que contiene la página). Un bloque de borrado puede contener de varias docenas a más de 100 páginas.

Una excepción a la granularidad de páginas de lectura y de escritura anterior son escrituras de sub-página, o programación de página parcial. Dependiendo de la tecnología, las páginas pueden escribirse parcialmente hasta 4 veces antes de que se requiera un borrado.

Debido a que las páginas en un bloque de flash de NAND pueden escribirse de forma secuencial y solo una vez entre operaciones de borrado de bloque, las escrituras subsiguientes requieren una escritura en una página diferente, que por lo general está ubicada en un bloque de flash diferente. La cuestión de los borrados de bloque se maneja mediante la creación de una agrupación de bloques de flash grabables, una función del sistema de archivos de flash.

El borrado de un bloque de borrado es la operación más costosa desde el punto de vista del tiempo, debido a que esta puede llevar varios milisegundos. Para los dispositivos que están sometidos a un uso intensivo (desde el punto de vista del tráfico), la velocidad a la que pueden generarse bloques de borrado (es decir, con qué rapidez pueden facilitarse unos bloques de borrado libres) es a menudo un factor limitante en el diseño de flash.

Muchos SSD (discos de estado sólido) usan la tecnología de flash. El soporte lógico inalterable en el SSD maneja las cuestiones de acceso que se han mencionado en lo que antecede en una capa que se denomina capa de traducción de flash (FTL, *Flash Translation Layer*). Al hacer esto, no obstante, el soporte lógico inalterable realiza suposiciones acerca de cómo se usará el SSD (por ejemplo, en su mayor parte lecturas, en su mayor parte escrituras, tamaño y alineación de lecturas y escrituras), y como resultado de estas suposiciones, las características de rendimiento del SSD son a menudo sub-óptimas para los algoritmos de indexación.

Muchos algoritmos de indexación que pueden encontrarse en la literatura y en la práctica se basan en un modelo de acceso a memoria uniforme, es decir, puede accederse por igual a toda la memoria desde el punto de vista del tiempo tanto lecturas como para escrituras, y no hay restricción de primer orden alguna sobre el tamaño de acceso o la alineación.

Si se considera una solución de indexación, las operaciones tales como inserción, eliminación, consulta y modificación por lo general requieren unas cantidades de tiempo mayores y variadas, y las lecturas y escrituras de bloques, por lo general bloques pequeños (4 kB o así), menos tiempo. Los bloques parecen ser aleatorios, es decir, puede leerse cualquier bloque, y puede escribirse cualquier otro bloque. Con algunos algoritmos, hay perfiles de E / S de lectura aleatoria - modificación - escritura, es decir, un bloque aleatorio se lee y, a continuación, se reescribe en la misma ubicación con unos datos ligeramente modificados.

Esta E / S aleatoria que es necesario que un algoritmo de indexación opere de forma eficiente, no es lo que flash tiene por objeto proporcionar. A pesar de que flash puede manejar bien las lecturas aleatorias, las escrituras aleatorias son difíciles, como los son las lecturas - modificaciones - escrituras. La razón de esto es que no se puede sobreescribir algo que ya se ha escrito, ha de borrarse esto en primer lugar. Para complicar adicionalmente la situación, el borrado lleva tiempo, y ha de tener lugar en unas fronteras grandes (por lo general 64 kB).

Cuando se borra un bloque de borrado, es necesario que cualquier dato válido en ese bloque sea movido a otra parte. Si el algoritmo escribe unos bloques de 4 kB aleatorios a lo largo del dispositivo de flash, una implementación sin experiencia daría como resultado que se borraran bloques la totalidad del tiempo. Debido a que los tiempos de borrado son lentos, el rendimiento se resentiría de forma significativa.

De acuerdo con la invención, para permitir que las escrituras en el elemento de flash sean secuenciales, a la vez que se sigue conservando el acceso aleatorio lógico que espera el algoritmo de indexación, se crea una tabla de traducción o de direccionamiento indirecto. Esta tabla pone en correspondencia unos sectores de almacenamiento lógicos (de registros) según lo necesite el algoritmo de indexación con unos sectores de almacenamiento físicos (por ejemplo, páginas) del dispositivo de flash.

Debido a que el algoritmo de indexación coloca en memoria sectores de almacenamiento (por ejemplo, páginas de datos a partir de flash), con el fin de modificar los contenidos de sector de almacenamiento (operaciones de inserción, de actualización o de eliminación), los sectores de almacenamiento se mueven a una memoria caché. Los sectores de almacenamiento correspondientes en el dispositivo de flash pueden marcarse ahora como no válidos (libres). En el caso de un SSD, este puede adoptar la forma de una orden TRIM.

De acuerdo con realizaciones adicionales de la invención, se proporcionan métodos para generar unos bloques de borrado libres. En cualquier instante dado, un bloque de borrado puede tener una combinación de datos válidos y no válidos. Para liberar un bloque de borrado, todos los datos válidos han de sacarse de ese bloque. Hay dos mecanismos que pueden usarse para lograr esto. Uno es el uso de las lecturas aleatorias que se generan por el algoritmo de indexación para leer más (de lo que se requiere por el algoritmo de indexación) con el fin de liberar un bloque de borrado. Debido a que el algoritmo de indexación tiende a generar unas lecturas aleatorias, con el tiempo todos los bloques de borrado finalmente se leen y se recolectan en busca de páginas vacías. Por ejemplo, si el bloque de borrado que contiene la lectura tiene algunas páginas libres, y algunas páginas válidas, entonces el algoritmo puede elegir colocar en memoria la totalidad del bloque de borrado y colocar todas las páginas válidas en la memoria caché. Esto tiene el efecto de liberar ese bloque de borrado para un borrado subsiguiente y, a continuación, una escritura.

Como alternativa, por ejemplo, si el proceso de lectura aleatoria que se ha mencionado en lo que antecede no es lo bastante rápido, puede usarse un proceso de depuración separado (por ejemplo, un subproceso) para leer bloques de borrado, y colocar las páginas válidas en la memoria caché para una fusión en otro bloque de borrado.

A medida que se llena la memoria caché, las entradas han de escribirse. Se recopila un conjunto de entradas de memoria caché que se escribirá de forma secuencial en un conjunto contiguo de páginas parciales (si se permiten escrituras de página parcial por el dispositivo de flash), múltiples páginas, y / o uno o más bloques de borrado. A medida que las entradas de memoria caché se escriben en el dispositivo de flash, la tabla de direccionamiento indirecto se actualiza, de tal modo que el algoritmo de indexación sigue viendo las entradas como que se encuentran en una dirección lógica fija.

4. Operaciones de indexación

Diversas realizaciones de la invención se describirán a continuación utilizando las figuras adjuntas 9 - 14 para ilustrar diversas operaciones de indexación que se realizan de acuerdo con la presente invención. Las figuras 15 - 16 ilustran dos métodos de generación de unos bloques de borrado libres para una utilización eficiente del medio de almacenamiento (por ejemplo, memoria flash). Estas realizaciones tienen por objeto ser ilustrativas y no limitantes.

La figura 9 es una visión de conjunto de varias operaciones de indexación que utilizan una tabla de traducción de sectores de almacenamiento 17' y una memoria caché 23' de acuerdo con una realización de la invención. En la parte superior de la figura 9, tres operaciones de índice 12' - 14' se muestran como unas entradas alternativas a una función de consulta 15' y una función de traducción 16'. Una primera operación de índice 12' es "clave de consulta" para devolver datos de satélite a partir de (una entrada de registro) para la clave. Una segunda operación de índice 13' es "actualizar datos de satélite para clave" para actualizar (modificar) la entrada de registro para la clave. Una tercera operación de índice 14' es "insertar nueva clave" para insertar una nueva entrada de registro. Otra operación de índice, eliminación, no se muestra en la figura 9 sino que se describe en lo sucesivo con respecto a la figura 13.

La totalidad de las tres operaciones de índice realizan en primer lugar una función de consulta 15', donde alguna función de la clave f(clave) se usa para generar un índice, en el presente caso un identificador de sector de almacenamiento lógico que soporta (por ejemplo, acelera) una consulta en tabla de troceo. El identificador de sector de almacenamiento (índice) se introduce en una función de traducción 16' donde alguna función del identificador de sector de almacenamiento lógico f(índice) genera una ubicación de sector de almacenamiento físico en la memoria flash. La función de traducción se implementa mediante una tabla de traducción de sectores de almacenamiento 17', que es una correspondencia del identificador de sector de almacenamiento lógico (tal como se proporciona por el algoritmo de indexación) con una ubicación objetivo de la memoria flash (ubicación de sector de almacenamiento físico en flash). Un diccionario (índice) almacenado en la memoria flash 26' puede comprender unos registros que ponen en correspondencia una clave de consulta (por ejemplo, un nombre de objeto) con datos de satélite (por ejemplo, un puntero de ubicación al objeto que está almacenado en disco).

A continuación, dependiendo de cuál de las tres operaciones de indexación se está realizando (consulta, actualización o inserción), se realizan una o más de las etapas que se muestran en la mitad inferior de la figura 9.

Para una operación de consulta 18', la entrada de sector de almacenamiento que se identifica mediante la función de traducción se lee 30' a partir del sector de almacenamiento 22' objetivo en memoria flash, con un vistazo anticipado a la memoria caché (por ejemplo, si el sector de almacenamiento objetivo se almacena en una memoria caché, este puede leerse a partir de la memoria caché 23' en lugar de a partir de la memoria flash 26').

Para una operación de actualización 19', la entrada de sector de almacenamiento que se identifica mediante la función de traducción (la entrada de sector de almacenamiento original) se lee 30' a partir de un sector de almacenamiento 22' objetivo en el bloque de borrado 21 a' de memoria flash (o memoria caché), el sector de almacenamiento se actualiza y se mueve 32' a memoria caché, y en una escritura subsiguiente 24' una pluralidad de entradas de sector de almacenamiento de memoria caché se leen de forma secuencial a un conjunto contiguo de páginas parciales, múltiples páginas y / o bloques de borrado (por ejemplo, un nuevo bloque de borrado 21 b') en memoria flash. El proceso actualiza 33' el estatus de la totalidad de los sectores de almacenamiento movidos en flash a datos no válidos (por ejemplo, libres o disponibles para una operación de recorte).

Para una operación de inserción 20', un sector de almacenamiento objetivo se lee de nuevo a partir de flash y una entrada de sector de almacenamiento modificado es movida 34' a memoria caché, de nuevo para una escritura secuencial subsiguiente 24' en una nueva ubicación en memoria flash.

La figura 9 muestra de forma esquemática una memoria caché 23' para recopilar una pluralidad de entradas de sector de almacenamiento, antes de realizar una escritura secuencial 24' de la recopilación de entradas de sector de almacenamiento de memoria caché en unos sectores de almacenamiento movidos de memoria flash contiguos. En una realización, una operación de depuración 25' se usa para crear unos bloques de borrado libres; el proceso incluye almacenar cualesquiera sectores de almacenamiento válidos (a partir del bloque de borrado) en una memoria caché durante el proceso de depuración y la reasignación del bloque de borrado de flash como libre.

Siguiendo un análisis de las nuevas estructuras de datos que se ilustran en la figura 10, las operaciones de indexación a las que se hace referencia en la figura 9 se describirán más específicamente con respecto a los diagramas de flujo de las figuras 11 - 14.

5. Estructuras de datos

La figura 10 ilustra diversas realizaciones de estructuras de datos útiles en la presente invención. Tales estructuras de datos tienen por objeto ser ilustrativas y no limitantes.

La figura 10A ilustra una realización de una tabla de traducción de sectores de almacenamiento (BTT, *bucket translation table*) 300' para traducir un índice de sectores de almacenamiento lógicos (que se genera por el algoritmo de indexación) a una dirección de sector de almacenamiento de flash física. Se muestra una entrada de tabla BTT que tiene tres campos: válido 301'; dirección de sector de almacenamiento físico de flash 302'; y el estado de sector de almacenamiento ampliado 303'. La granularidad de las direcciones de sector de almacenamiento es el tamaño de escritura mínimo del dispositivo de flash, en concreto o bien una escritura de página parcial (por ejemplo, para NAND de SLC) o bien una escritura de página (por ejemplo, para NAND de MLC). La BTT es una puesta en correspondencia 1 : 1 de entradas de sector de almacenamiento lógicas y físicas. La tabla posibilita la reorganización de las asignaciones de sectores de almacenamiento de flash para un rendimiento aleatorio (lecturas aleatorias y escrituras aleatorias por el algoritmo de indexación) más alto. Una información de estado adicional puede añadirse a la BTT en el tercer campo para posibilitar la aceleración de algoritmos.

La figura 10B muestra una realización de una tabla de sectores de almacenamiento válidos (BVT, *bucket valid table*) 305'. Esta tabla realiza un seguimiento de qué sectores de almacenamiento físicos en flash son válidos con el fin de gestionar la depuración de sectores de almacenamiento para dar bloques para su recorte. Como un ejemplo, un campo 306' etiquetado como válido puede ser una disposición de bits compacta (1 bit / sector de almacenamiento). El tamaño de la BVT es el número total de entradas de sector de almacenamiento de flash, solo un subconjunto de las cuales se encuentran en uso por la BTT.

La figura 10C ilustra una realización del sector de almacenamiento de flash 309' que tiene múltiples registros 310', 311', 312' ... incluidos en el sector de almacenamiento, junto con un puntero de BTT inverso 313' (un auto-índice en la tabla de traducción de sectores de almacenamiento 17'). Por lo tanto, cada sector de almacenamiento contiene un conjunto de uno o más registros y un puntero inverso para actualizar la BTT cuando se insertan, se mueven o se eliminan unos sectores de almacenamiento de flash (por ejemplo, páginas). Cada elemento del sector de almacenamiento (registro o puntero) puede tener un contenido redundante añadido, tal como bits de ECC adicionales, para mejorar la fiabilidad individual de las estructuras de datos y aumentar de forma significativa la vida útil de los dispositivos de almacenamiento. Por ejemplo, un campo de número de secuencia opcional puede añadirse al sector de almacenamiento de flash 309' para realizar una comprobación de consistencia de los datos durante sucesos de fallo de alimentación; también pueden proporcionarse otras banderas de optimización.

Debido a que el tamaño de registro es pequeño en relación con el tamaño de sector de almacenamiento, esto proporciona una oportunidad (opcional) de implementar una información de recuperación en casos de error adicional en función de los registros individuales. Esta característica opcional mejoraría la fiabilidad en conjunto de la solución mediante el aumento del número de errores de bits y fallos que pueden corregirse y por lo tanto aumentar la vida operativa efectiva de la tecnología de almacenamiento subyacente.

La figura 10D muestra un ejemplo de un dispositivo de flash de NAND de SLC 315' que contiene múltiples bloques de borrado 316' (1 a M). Cada bloque de borrado incluye múltiples páginas 317' (1 a N). En este ejemplo, cada página es de 4 kB y cada página incluye múltiples sectores de almacenamiento 318' (1 a B), siendo cada sector de almacenamiento de 1 kB. En este ejemplo, el dispositivo soporta escrituras de página parcial.

Un sector de almacenamiento representa un tamaño de escritura mínimo del dispositivo de flash. Por lo general, un sector de almacenamiento sería una página. Si se permiten escrituras de página parcial, entonces pueden proporcionarse uno o más sectores de almacenamiento por página de flash, tal como un dispositivo de NAND de SLC de cuatro páginas parciales que soporta cuatro sectores de almacenamiento por página.

Se proporcionan múltiples páginas de flash por bloque de borrado. Hay múltiples bloques de borrado por dispositivo de flash, y cada bloque se borra de forma individual.

El subsistema de flash típico consiste en múltiples dispositivos de flash. Se escribe en los dispositivos de flash de NAND de forma secuencial una vez por página (o página parcial) dentro de un bloque dado entre operaciones de borrado, con múltiples bloques disponibles para escribir y leer de forma simultánea.

6. Diagramas de flujo de proceso

La figura 11 ilustra una realización de un proceso de operación de consulta para verificar la presencia de una clave y devolver datos de satélite asociados. En la etapa uno 41', una clave de consulta se introduce en una función de consulta. En la etapa dos 42', la función de consulta f(clave) genera un identificador de sector de almacenamiento lógico que soporta (por ejemplo, acelera) una consulta en tabla de troceo. El identificador de sector de almacenamiento lógico se introduce en una función de traducción, que en la etapa tres 43' se pone en correspondencia con una ubicación (de sector de almacenamiento físico) de memoria flash, por medio de la tabla de traducción de sectores de almacenamiento (BTT) 17'. En la etapa cuatro 44', el sector de almacenamiento objetivo en memoria flash se lee 45a' a partir de memoria flash, a menos que el sector de almacenamiento esté almacenado en una memoria caché, caso en el cual este puede leerse 45b' a partir de la memoria caché 23'. En la etapa seis 46', los datos (de registro) de satélite para la clave se devuelven al algoritmo de indexación.

La figura 12 muestra una realización de un proceso de operación de inserción. Una primera etapa 71' introduce una clave en la función de consulta. En la etapa dos 72', la función de consulta f(clave) genera un índice, en el presente caso un identificador de sector de almacenamiento lógico. En la etapa tres 73', el identificador de sector de almacenamiento se introduce en una función de traducción que pone en correspondencia el identificador de sector de almacenamiento con una ubicación de sector de almacenamiento físico de memoria flash donde debería tener lugar la inserción, utilizando la tabla de traducción de sectores de almacenamiento (BTT) 17'. En la etapa cuatro 74', el proceso de inserción recibe la ubicación de sector de almacenamiento objetivo a partir de la función de traducción. En la etapa cinco 75', el proceso de inserción lee el sector de almacenamiento 22' objetivo a partir de un bloque de borrado 21a' de la memoria flash 75a', o a partir de la memoria caché 75b'. En la etapa seis 76', el proceso de inserción inserta la entrada de registro en el sector de almacenamiento objetivo y escribe el sector de almacenamiento modificado en una memoria caché. En la etapa siete 77', múltiples entradas de sector de almacenamiento (incluyendo el sector de almacenamiento objetivo modificado) se leen a partir de la memoria caché 73' por el proceso de inserción. En la etapa ocho 78', el proceso de inserción escribe el sector de almacenamiento objetivo modificado y otros sectores de almacenamiento que se leen a partir de una memoria caché en nuevas ubicaciones (páginas en el bloque de borrado 21 b') en la flash 26'. En la etapa nueve 79', el proceso de inserción actualiza la tabla de traducción de sectores de almacenamiento 17' con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos de memoria caché a la memoria flash 79a', y también actualiza las entradas válidas de sector de almacenamiento en la BVT 79b' para todos los sectores de almacenamiento que son movidos. En la etapa diez 80', el proceso de inserción marca las entradas de memoria caché movidas de solo lectura (disponibles). En la etapa once 81', el proceso de inserción marca los sectores de almacenamiento de flash originales (movidos ahora a un nuevo bloque de borrado) como libres.

La figura 13 ilustra una realización de un proceso de operación de eliminación. En una primera etapa 91', se proporciona una clave a una función de consulta. En la etapa dos 92', la función de consulta f(clave) genera un índice, en el presente caso un identificador de sector de almacenamiento lógico. En la etapa tres 93', el identificador de sector de almacenamiento se proporciona a la función de traducción, que utiliza la tabla de traducción de sectores de almacenamiento 17' para poner en correspondencia el identificador de sector de almacenamiento con una ubicación de sector de almacenamiento de memoria flash física. En la etapa cuatro 94', el proceso de eliminación recibe la ubicación de la memoria flash. En la etapa cinco 95', el sector de almacenamiento objetivo se lee a partir de flash. En la etapa seis 96', el proceso elimina la entrada de registro original en el sector de almacenamiento y escribe el sector de almacenamiento modificado (con la entrada eliminada) en la memoria caché 23'. En la etapa siete 97', un grupo (recopilación) de sectores de almacenamiento se leen a partir de una memoria caché. En la etapa ocho 98', el sector de almacenamiento objetivo actualizado y otros sectores de almacenamiento que se leen a partir de la memoria caché 23' se escriben de forma secuencial en un conjunto contiguo de páginas libres en flash. En la etapa nueve 99', el proceso de eliminación actualiza la tabla de traducción de sectores de almacenamiento con las nuevas ubicaciones en flash para todos los sectores de almacenamiento movidos 99a', y actualiza su estatus válido en la BVT 99b'. En la etapa diez 100', el proceso de eliminación marca las entradas de memoria caché como de solo lectura. En la etapa once 101', el proceso de eliminación marca los sectores de almacenamiento de flash originales movidos ahora a una nueva ubicación en flash como libres.

La figura 14 ilustra una realización de un proceso de operación de actualización para modificar un registro en un índice que está almacenado en memoria flash. En una primera etapa 51', una clave se proporciona como entrada a una función de consulta. En la etapa dos 52', la función de consulta f(clave) genera un índice, en el presente caso un identificador de sector de almacenamiento lógico. El identificador de sector de almacenamiento se introduce en una función de traducción. En la etapa tres 53', la función de traducción pone en correspondencia el identificador de sector de almacenamiento con un sector de almacenamiento físico en memoria flash donde debería tener lugar la inserción, utilizando la tabla de traducción de sectores de almacenamiento 17'. En la etapa cinco 55', el sector de almacenamiento objetivo se lee a partir de la memoria flash 55a' o a partir de la memoria caché 55b'. En la etapa seis 56', después de actualizar la entrada, el sector de almacenamiento actualizado se escribe en la memoria caché 23'. En la etapa siete 57', un grupo de sectores de almacenamiento se leen a partir de la memoria caché 23' y en una etapa ocho 58', se escriben de forma secuencial a partir de una memoria caché en una nueva ubicación en la memoria flash 26'. En la etapa nueve 59', el proceso de actualización actualiza la tabla de traducción de sectores de almacenamiento 17' con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos 59a', y actualiza su estatus válido en la BVT 59b'. En la etapa diez 60', el proceso de actualización marca las entradas movidas como de solo lectura en la memoria caché 23' (y por lo tanto disponibles para que se escriba sobre las mismas). Por último, en la etapa once 61', el proceso de actualización marca los sectores de almacenamiento de flash originales, movidos ahora a una nueva ubicación, como libres (disponibles).

La figura 15A ilustra una realización de un proceso para generar unos bloques de borrado libres, donde una lectura a demanda (que se genera por una operación de indexación de aguas arriba tal como una consulta, inserción o modificación) lee unos sectores de almacenamiento adicionales en el mismo bloque de borrado (que el sector de almacenamiento objetivo). En la figura 15A, el proceso se ilustra con una solicitud de actualización. En la etapa uno 111', una clave se proporciona a una función de consulta. En la etapa dos 112', la función de consulta f(clave) genera un índice, en el presente caso un identificador de sector de almacenamiento lógico. En la etapa tres 113', el identificador de sector de almacenamiento se pone en correspondencia con una ubicación de sector de almacenamiento objetivo física en flash. En la etapa cuatro 114', el proceso de actualización y de depuración recibe

la ubicación objetivo de la memoria flash. En la etapa cinco 115', el proceso identifica todos los sectores de almacenamiento válidos en el mismo bloque de borrado que el sector de almacenamiento objetivo. En la etapa seis, 116', el proceso de actualización lee el sector de almacenamiento objetivo y todos los sectores de almacenamiento válidos identificados a partir del bloque de flash que contiene el sector de almacenamiento objetivo. En la etapa siete 117', el proceso actualiza la entrada de registro en el sector de almacenamiento objetivo y escribe todos los sectores de almacenamiento válidos a partir del bloque de flash en la memoria caché 23'. En la etapa ocho 118', el proceso de actualización lee un grupo de bloques a partir de una memoria caché. En la etapa nueve 119', el proceso de actualización escribe el sector de almacenamiento objetivo actualizado y otros sectores de almacenamiento que se leen a partir de la memoria caché 23' en la flash 26'. En la etapa diez 120', el proceso de actualización actualiza la tabla de traducción de sectores de almacenamiento 17' con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos (se escriben a partir de una memoria caché en el nuevo bloque de borrado 21 b' en flash) 120a', y actualiza las entradas de sector de almacenamiento en la BVT 120b'. En la etapa once 121', el proceso de actualización marca las entradas de memoria caché, ahora obsoletas, como de solo lectura. En la etapa doce 122', el proceso de actualización marca el bloque de flash original (todos los sectores de almacenamiento en el bloque objetivo) como libre.

La figura 15B ilustra una realización particular del proceso de lectura aleatoria que acaba de describirse para generar unos bloques de borrado libres.

En la presente realización, un algoritmo de indexación de aplicación de troceo de desplazamiento 125' genera unos sectores de almacenamiento lógicos 126'. El tamaño de sector de almacenamiento lógico tal como es visto por el algoritmo de indexación, está vinculado al tamaño de bloque de borrado de flash con el fin de volver compatible el algoritmo de indexación y la memoria flash. Estos sectores de almacenamiento se leerán de forma aleatoria como resultado de las lecturas y actualizaciones de índice.

Una tabla de traducción de sectores de almacenamiento (direccionamiento indirecto) 127' traduce un índice de sectores de almacenamiento lógicos a una ubicación de sector de almacenamiento de dispositivo de flash físico. Esta tabla de direccionamiento indirecto posibilita que el algoritmo de indexación trabaje de forma aleatoria, para lecturas, escrituras y actualizaciones, y aún tenga grandes escrituras secuenciales que se realizan al nivel de dispositivo de flash. Preferiblemente, la tabla de direccionamiento indirecto se almacena en una memoria persistente, pero esta puede reconstruirse según sea necesario si se almacena en memoria volátil.

La salida de la tabla de direccionamiento indirecto, en concreto la ubicación de sector de almacenamiento de dispositivo físico, se proporciona como entrada a una memoria caché de sectores de almacenamiento completamente asociativa 128'. En la presente realización, si los contenidos de una fifo de bloque de borrado vacía 129' se encuentran por debajo de una marca de cota máxima Q, entonces se lee la totalidad del bloque de borrado (que contiene el sector de almacenamiento de 4 kB objetivo).

Los bloques de borrado hospedan unos sectores de almacenamiento lógicos, siendo una configuración típica un bloque de borrado que contiene 16 de los sectores de almacenamiento lógicos de 4 kB. El dispositivo físico está configurado para una carga, por ejemplo, de un 90 %, lo que quiere decir que un 90 % de los sectores de almacenamiento se encuentran en uso. El almacenamiento en memoria caché y el descarte (expulsión) se usan para agrupar (concentrar) sectores de almacenamiento lógicos en la memoria flash de tal modo que la mayor parte del 10 % de los sectores de almacenamiento restantes está concentrada en unos bloques de borrado libres.

El descarte de memoria caché (proceso de expulsión) toma 16 sectores de almacenamiento, recopilados en una memoria caché, y escribe los 16 sectores de almacenamiento a partir de una memoria caché en un bloque de borrado libre 130'. Debido a que los bloques de borrado son tocados de forma aleatoria por las operaciones de lectura aleatoria, las operaciones de lectura pueden usarse para generar unos bloques de borrado libres. El uso de una función de troceo criptográfico para generar los identificadores de sector de almacenamiento lógico, aumentará la naturaleza aleatoria de las operaciones de lectura y por lo tanto mejorará la generación de lectura aleatoria de bloques de borrado libres.

Las figuras 16A y 16B ilustran un proceso de depuración alternativo para generar unos bloques de borrado libres. Este proceso de depuración no es una parte de operación de indexación alguna. Más bien, este se implementa como parte de una capa de gestión de dispositivo de nivel más bajo. En este proceso, un grupo (algunos o todos) de los sectores de almacenamiento físicos en un bloque de borrado de flash se leen directamente a partir de flash y la tabla de sectores de almacenamiento válidos 27' se usa para determinar qué sectores de almacenamiento en el bloque de borrado son válidos.

Tal como se ilustra en la figura 16A, en la etapa uno 220', un proceso de depuración 25' lee un bloque de borrado 21a' completo. En la etapa dos 222', el proceso de depuración usa la tabla de sectores de almacenamiento válidos 27' para identificar todos los sectores de almacenamiento de aquellos leídos que son válidos. En la etapa tres 224', para cada sector de almacenamiento válido, el identificador de sector de almacenamiento lógico se extrae del sector de almacenamiento. En la etapa cuatro 226', los sectores de almacenamiento válidos se almacenan en la memoria caché 23', cada uno indexado por su identificador de sector de almacenamiento lógico.

La figura 16B muestra un ejemplo donde en la etapa uno, el proceso de depuración 25' lee los sectores de almacenamiento [94, 97] inclusive. En la etapa dos, el proceso determina que los sectores de almacenamiento en 95 y 96 son válidos. Los sectores de almacenamiento válidos se muestran en la tabla de sectores de almacenamiento válidos designados por un "1", y los sectores de almacenamiento no válidos por un "0". En la etapa tres, los identificadores de sector de almacenamiento lógico para los sectores de almacenamiento 95 y 96, en concreto las etiquetas 23 y 49 respectivamente, se extraen de los sectores de almacenamiento. En la etapa cuatro, las dos etiquetas, y sus sectores de almacenamiento 95 y 96 respectivos se insertan en una memoria caché usando sus etiquetas 23, 49 respectivas como el índice.

7. Vista e implementación de nivel de pila

Otro ejemplo más específico de la invención se describirá a continuación con respecto a las figuras 17 - 24.

La figura 17 muestra una vista o pila de seis capas 200' para ilustrar una implementación de la presente invención donde una capa de adaptación de flash 207' adapta una vista de perfil de uso de E / S que es deseada por un algoritmo de indexación 203, que es una vista muy diferente de la deseada por el dispositivo de memoria flash físico 211'. En el nivel de arriba 201', se proporciona un diccionario (índice) de registros, para el cual se requieren determinadas operaciones de indexación 204' (consulta, eliminación, inserción y modificación de un registro). Una capa de algoritmo de indexación 203' implementa el diccionario con uno o más algoritmos de indexación, por ejemplo, siendo un ejemplo un algoritmo de aplicación de troceo de desplazamiento de cuco. El algoritmo de indexación tiene una vista de cómo se almacenarán las claves en el índice mediante una capa de persistencia de índices 205'. La vista de indexación es una vista lógica, que especifica ubicaciones de dirección lógica. La vista supone adicionalmente que habrá un acceso uniforme al índice con respecto al tamaño, la alineación y la temporización, y que el índice se almacena en un almacenamiento mutable (estable).

La capa de persistencia de índices 205' presentará unas operaciones con sectores de almacenamiento lógicos 206' para leer y escribir, en unos sectores de almacenamiento físicos que almacenan los registros del índice. Estas operaciones con sectores de almacenamiento lógicos 206' se presentan a una capa de adaptación de flash 207' que, tal como se ha descrito previamente, traduce los sectores de almacenamiento lógicos (del proceso de indexación) a unas ubicaciones de sector de almacenamiento físico en el dispositivo de almacenamiento de flash. La capa de adaptación de flash por lo tanto adapta la vista y el perfil de uso de E / S que son deseados por el algoritmo de indexación en lo que antecede, a la vista muy diferente deseada por el dispositivo de almacenamiento físico (la memoria flash 211') en lo sucesivo. En el presente caso las operaciones con sectores de almacenamiento físicos 208' incluyen unas lecturas aleatorias y unas escrituras agregadas (secuenciales en bloque), que constituyen un modelo no uniforme de acceso a sectores de almacenamiento. Las operaciones con sectores de almacenamiento físicos en este ejemplo pueden incluir además unas órdenes de recorte.

Las operaciones con sectores de almacenamiento físicos se implementan mediante una capa de gestión de dispositivo 209' que realiza un seguimiento de y coordina los recursos en el dispositivo de flash físico. Estas operaciones de dispositivo físico 210' en el presente caso incluyen unas lecturas aleatorias, grandes escrituras secuenciales y unas órdenes de recorte.

La capa de dispositivo físico 211' está caracterizada por su lectura y escritura no uniforme e inmutabilidad con respecto al tamaño, la alineación y la temporización. Los ejemplos de tales dispositivos físicos incluyen flash en bruto, de cambio de fase, un SSD, y / o flash con un sistema de archivos de flash que reside en el dispositivo.

La presente invención posibilita unas potenciaciones opcionales adicionales por debajo de la capa de gestión de dispositivo tal como:

- El modelo de recorte de sectores de almacenamiento (recorte fino de páginas) y de seguimiento de sectores de almacenamiento dentro de una página posibilita una mejor gestión de bloques de borrado si se incorpora directamente a un sistema de archivos de flash de un dispositivo de almacenamiento de SSD o equivalente.
- La puesta en correspondencia de sectores de almacenamiento con páginas de flash es una abstracción. Los sectores de almacenamiento podrían ponerse en correspondencia con páginas parciales, para NAND de SLC para aumentar la vida de esos dispositivos mediante la minimización de la cantidad de datos que se escriben en el elemento de flash para cada cambio. Los sectores de almacenamiento también pueden ponerse en correspondencia con múltiples páginas de flash si esto fuera beneficioso para el rendimiento del sistema en conjunto.

La figura 18 muestra un ejemplo de un registro de índices. El registro 140' es de 32 bytes en total, incluyendo un primer campo de 20 bytes 141' para almacenar una huella digital (clave). Una huella digital es preferiblemente un compendio de troceo criptográfico del contenido de datos, por ejemplo, un algoritmo de troceo SHA-1. Para facilitar la ilustración, en lugar de teclear la huella digital en dígitos en hexadecimal tales como "AB92345E203 ..." se designará una huella digital individual en las figuras 19 - 22 mediante una única letra mayúscula tal como P, Q, R, S, T. Estas letras mayúsculas también actuarán como un representante para la totalidad del registro, de nuevo para simplificar con fines de ilustración. Los campos del registro también incluyen un campo de recuento de referencia de

dos bytes 142', un campo de dirección de bloque físico de cinco bytes 143', un campo de banderas de un byte 144', y un campo de miscelánea de cuatro bytes 145'. El campo de PBA 143' contiene un puntero a la dirección de bloque físico de los datos almacenados en disco, para la huella digital 141' designada. El recuento de referencia realiza un seguimiento del número de referencias a los datos almacenados en disco.

De acuerdo con una realización de la invención, la huella digital 141' a partir del registro de índices se usa como una clave de entrada para la función de consulta f(clave) que se ha descrito previamente (la figura 9), En este ejemplo, la función f(clave) comprende un conjunto de cuatro funciones de troceo H_o, H_1, H_2, y H_3. En general, puede usarse cualquier conjunto de dos o más funciones de troceo. La función de troceo H_x pone en correspondencia la huella digital con un intervalo [0, N - 1] inclusive, donde N es el tamaño de la tabla de troceos. Dado que en este ejemplo las propias huellas digitales son troceos, se pueden extraer campos de bits para generar la siguiente familia de cuatro valores de troceo:

$H_0(x) = x<0:31> \bmod N$
$H_1(x) = x<032:63> \bmod N$
$H_2(x) = x<064:95> \bmod N$
$H_3(x) = x<096:127> \bmod N$

La anchura del campo de bits que se extrae es mayor que o igual a $\log_2 (N)$. Puede usarse cualquier combinación de bits inconexos, sometidos a la restricción de $\log_2 (N)$. Tal como se ilustra en la figura 18, solo la huella digital en el primer campo 141' se somete a troceo, para formar la clave. El contenido restante (los campos 142' - 145') del registro 140' comprende un valor o una cabida útil.

La figura 19 ilustra un ejemplo de un algoritmo de indexación de aplicación de troceo de desplazamiento que se conoce como aplicación de troceo de cuco. Para facilitar la ilustración, se usan solo dos funciones. La figura 19A muestra una cuadrícula de 2 x 3 donde la huella digital P genera los valores de troceo 2 y 5 a partir de las funciones $H_0(x)$ y $H_1(x)$, respectivamente, mientras que la huella digital Q genera los valores de troceo 1 y 3 a partir de estas mismas funciones. El algoritmo de aplicación de troceo de cuco realizará una selección de entre los dos valores de troceo alternativos para colocar P y Q en una de las siete ranuras etiquetadas 0 - 6 (la figura 19B). P puede ir en una de dos ubicaciones, 2 o 5, y Q puede ir en una de dos ubicaciones, 1 o 3. El algoritmo pone Q en la ranura vacía más baja 1 y P en la ranura 2, tal como se muestra en la figura 19C. A pesar de que en este ejemplo se hace referencia al contenedor de registros como una ranura que contiene un registro, debería entenderse que la invención no es tan limitada; los algoritmos de indexación también ven un sector de almacenamiento, que contiene múltiples registros, como un contenedor. En el presente caso se usa una ranura de único registro para simplificar la explicación.

A continuación, se proporciona otra huella digital R que genera los valores de troceo de 1 y 2 a partir de las mismas funciones de troceo (véase la tabla en la figura 19D). El algoritmo de aplicación de troceo colocará R en la ubicación a la izquierda, en concreto la ranura 1, desplazando la entrada actual Q (la figura 19E). Q se moverá ahora a la otra ubicación opcional que se especifica por $H_1(Q)$, en concreto la ubicación 3. El algoritmo seguirá desplazando registros hasta que cada registro acaba en una ranura vacía.

En este ejemplo, para lograr la operación de "inserción R", el algoritmo de indexación genera las siguientes solicitudes de lectura y de escritura:

lectura 1 (obtiene Q)
lectura 2 (obtiene P)
escritura 1 (escribir R)
lectura 3 (comprobación de validez)
escritura 3 (Q)

Las primeras dos lecturas se usan para validar que R no se encuentra ya presente en el índice. La comprobación de validez (lectura 3) determina si el número de ranura 3 está vacío; de ser así, entonces Q puede escribirse en la ranura 3 y el algoritmo se realiza debido a que no volvió a escribirse entrada alguna en la ranura 3. Si la ranura 3 no estaba vacía, entonces será necesario que la entrada actual en la ranura 3 se mueva a otra ranura. Los contenidos de la ranura 3 se conocen si se tiene una correspondencia de bits; de lo contrario, es necesario que se lea la entrada en la ranura 3 para determinar su estatus. Cada entrada contiene un bit válido que indica si esa entrada es válida. Válida quiere decir que esta se encuentra en uso (y la ocupante actual de la ubicación ha de ser desplazada). No válida quiere decir que la ubicación está vacía, y el registro que se está procesando puede escribirse allí. Los contenidos de los bits válidos también pueden almacenarse en una correspondencia de bits separada, a costa de algo de memoria.

El algoritmo de aplicación de troceo de cuco es recursivo, por que este sigue escribiendo sobre entradas, desplazando el contenido previo, hasta que este acaba en una entrada vacía. En la práctica, este proceso rara vez supera un desplazamiento.

El algoritmo de indexación tiene unas operaciones de registro tanto de sector de almacenamiento como individual. El algoritmo de indexación se ha descrito en lo que antecede (en la figura 19) como la colocación de un registro en un contenedor (ranura), pero se entiende por el algoritmo de indexación que los registros también pueden agregarse para dar sectores de almacenamiento, es decir, sectores de almacenamiento que contienen múltiples registros. Por lo tanto, el ejemplo anterior es no limitante y tiene por objeto ilustrar en general unas operaciones de registro.

Tal como se ha descrito previamente, debido a que la lectura y la escritura de registros individuales no es eficiente para la memoria flash, los registros individuales se agregan para dar sectores de almacenamiento. La figura 20 ilustra cuatro sectores de almacenamiento de este tipo, conteniendo cada uno dos o más registros, es decir, el sector de almacenamiento B_0 con las ubicaciones de registro 0 y 1, B_1 con las ubicaciones de registro 2 y 3, B_2 con las ubicaciones de registro 4 y 5, y B_3 con las ubicaciones de registro 6 y x. El tamaño de sector de almacenamiento es una función de (y preferiblemente es igual a) el tamaño de escritura mínimo que es indicado por el dispositivo de flash, es decir, o bien una escritura de página completa o bien una escritura de página parcial. Un tamaño de sector de almacenamiento típico puede ser de 4 kB. No se requiere ordenación específica alguna de los registros dentro del sector de almacenamiento - - la totalidad del sector de almacenamiento se examina en busca de un registro válido durante la operación de consulta, de tal modo que el registro podría insertarse en cualquier punto dentro del sector de almacenamiento. Cuando se desplaza, de acuerdo con el algoritmo de aplicación de troceo de cuco, una entrada en el sector de almacenamiento puede ser desplazada de manera aleatoria. El algoritmo de indexación por lo tanto escribe sectores de almacenamiento lógicos en lo que parecen ser unas ubicaciones aleatorias, uno de cada vez, que finalmente son agregadas por la capa de adaptación de flash para dar unas escrituras físicamente contiguas (secuenciales) en el dispositivo de flash.

La figura 21 ilustra un ejemplo de una entrada de sector de almacenamiento 160'. Un tamaño de sector de almacenamiento de 4 kB se basa en el tamaño de escritura mínimo de dispositivo subyacente, en el presente caso una página de 4 kB. El sector de almacenamiento de 4 kB incluye un primer campo de 4 bytes 161' que especifica el número de registros en la entrada de sector de almacenamiento. Un campo de etiqueta de 4 bytes 162' especifica el identificador de sector de almacenamiento lógico. Este identificador (etiqueta) es una dirección lógica, no una física. La tabla de traducción pone en correspondencia la dirección de sector de almacenamiento de algoritmo (ABA, *algorithm bucket address*) con una dirección de sector de almacenamiento de flash FBA. La memoria caché funciona como una memoria caché virtual (en la terminología de CPU), con cada línea de memoria caché (entrada) identificada mediante una etiqueta, una ABA en este caso. Cuando el algoritmo solicita registros, todo lo que sabe al atravesar la memoria caché es que la ABA solicitada se almacena en memoria caché; en dónde esta se pone en correspondencia con (la FBA) se encuentra en el extremo inferior de la memoria caché (por ejemplo, véase el puntero inverso 313' a la BTT, en la figura 10C). El sector de almacenamiento incluye el campo 163' para contener una pluralidad de registros R_0, R_1, R_2 ... , siendo cada registro de un tamaño de 32 bytes. En este ejemplo, un sector de almacenamiento de 4 kB contendrá: (4096 - 4 - 4) / 32 registros, es decir, aproximadamente 127 registros por sector de almacenamiento.

La figura 22 es un diagrama esquemático de un dispositivo de memoria flash 164' que ilustra los tamaños relativos de un sector de almacenamiento, una página y un bloque de borrado en una realización. El dispositivo de flash físico es un chip (paquete) 165' que es de de un tamaño de 2 GB. En el chip, hay dos pastillas (obleas de silicio) 166a', 167b'. En cada pastilla, puede haber 2^14 bloques de borrado, siendo cada bloque de borrado 167' por lo general de 64 kB. Una página 168' es el tamaño mínimo que puede escribirse, en el presente caso 4 kB, y determina el tamaño del sector de almacenamiento 169', también 4 kB, tal como se usa a mayor altura en la pila (véase la figura 17).

La figura 23 ilustra componentes seleccionados, de acuerdo con una realización de una capa de gestión de dispositivo (209' en la figura 17), para realizar un seguimiento de, y coordinar, los recursos en el dispositivo de flash físico. La figura 23A muestra (en la parte superior) una pluralidad de páginas (sectores de almacenamiento) 170', seguidas por una correspondencia de asignación de páginas 171' que indica qué páginas son válidas (1 es válida, 0 no es válida). A continuación de esto se encuentra una correspondencia de recortes pendientes 172', de páginas que van a recortarse en el futuro, pero a las que aún no se les ha realizado. Las correspondencias de asignación de páginas y de recortes pendientes pueden usarse en diversas realizaciones de la invención tal como se ha descrito previamente, para determinar si un sector de almacenamiento contiene unos datos válidos (véase la tabla de sectores de almacenamiento válidos 27' que se ilustra en la figura 9).

La figura 23B ilustra un ejemplo de una tabla de descriptores de bloques de borrado 175', indexada por un índice de bloques de borrado. Cada entrada de descriptor de bloque de borrado 176' incluye una pluralidad de campos, incluyendo número de borrados 177', número de escrituras parciales 178', número de lecturas parciales 179', número de lecturas completas 180', número de escrituras completas 181, y número de errores 182'. Esta información puede usarse en la generación de unos bloques de borrado libres tal como se ha descrito previamente en diversas realizaciones de la invención.

REIVINDICACIONES

1. Un método de acceso a un archivo que tiene un nombre de archivo y almacenado como un objeto de archivo de un sistema de archivos en un almacén de objetos en uno o más dispositivos de almacenamiento en ordenador,
teniendo el almacén de objetos un índice de ubicaciones (204) de nombres de objeto y ubicaciones físicas de objeto para los objetos (P, Q, R) almacenados en el almacén de objetos, teniendo cada objeto como su nombre de objeto una huella digital de objeto globalmente única (H(p), H(q), H(r)) que se deriva del contenido del objeto,
teniendo el objeto de archivo (401) su propia huella digital de objeto de archivo, que comprende una tabla de puesta en correspondencia (402) de huellas digitales de objeto para objetos de datos del archivo almacenado en el almacén de objetos, estando la tabla de puesta en correspondencia indexada por un desplazamiento en el contenido del archivo para determinar qué objetos del archivo van a ser referenciados,
comprendiendo el método acceder a objetos en el almacén de objetos usando el índice de ubicaciones (204) y huellas digitales de objeto incluyendo:

acceder a un objeto de directorio (501), que tiene su propia huella digital de objeto, en el almacén de objetos usando la huella digital de objeto, comprendiendo el objeto de directorio una puesta en correspondencia de números de nodo - i con nombres de archivo;
examinar el objeto de directorio usando un nombre de archivo para obtener un número de nodo - i;
acceder a un objeto de correspondencia de nodos - i (502, 650), que tiene su propia huella digital de objeto, en el almacén de objetos usando la huella digital de objeto de correspondencia de nodos - i, comprendiendo el objeto de correspondencia de nodos - i una puesta en correspondencia de números de nodo - i de sistema de archivos y huellas digitales de objeto;
usar el objeto de correspondencia de nodos - i (502, 650) para traducir el número de nodo - i a partir del objeto de directorio a una huella digital de objeto de archivo;
acceder al objeto de archivo (401), que tiene su propia huella digital de objeto, en el almacén de objetos usando la huella digital de objeto de archivo, por lo que los objetos de datos (410, 411, 412) del archivo almacenado en el almacén de objetos se encuentran disponibles para un acceso por medio de la tabla de puesta en correspondencia (402).

2. El método de la reivindicación 1:

en el que, la huella digital del objeto de correspondencia de nodos - i comprende una instantánea del sistema de archivos.

3. El método de la reivindicación 1, que incluye:

publicar la huella digital de objeto de correspondencia de nodos - i en otro sistema informático.

4. El método de la reivindicación 1, que incluye:

realizar una recuperación en casos de desastre usando la huella digital de objeto de correspondencia de nodos - i como una instantánea del sistema de archivos.

5. El método de la reivindicación 1, donde el objeto de correspondencia de nodos - i (502, 650, 659) incluye una huella digital de objeto de un objeto de correspondencia de nodos - i previa.

6. El método de la reivindicación 5, donde:

las huellas digitales de objeto de correspondencia de nodos - i previa comprenden una historia de instantáneas del sistema de archivos.

7. El método de la reivindicación 1, donde:

la tabla de puesta en correspondencia de objetos de archivo (402) comprende una lista lineal, una estructura en árbol o una tabla de direccionamiento indirecto.

8. El método de la reivindicación 1, que incluye:

generar un registro de transacciones de actividad de objetos, incluyendo lecturas, escrituras, eliminaciones y actualizaciones de recuentos de referencia.

9. El método de la reivindicación 1, que incluye:

añadir, modificar o eliminar un objeto de datos del archivo y generar una nueva huella digital de objeto de archivo.

10. El método de la reivindicación 1, que incluye:

utilizar una aceleración de soporte físico (113) para realizar uno o más de denominación de objetos, compresión y cifrado.

11. El método de la reivindicación 1, que incluye:

mantener un recuento de referencia para cada objeto, y actualizar el recuento de referencia del objeto cuando se añaden o se eliminan referencias al objeto.

12. El método de la reivindicación 1, que incluye:

cuando se cambia el contenido de un objeto de archivo o un objeto de datos, propagar el cambio hacia arriba hasta un objeto raíz.

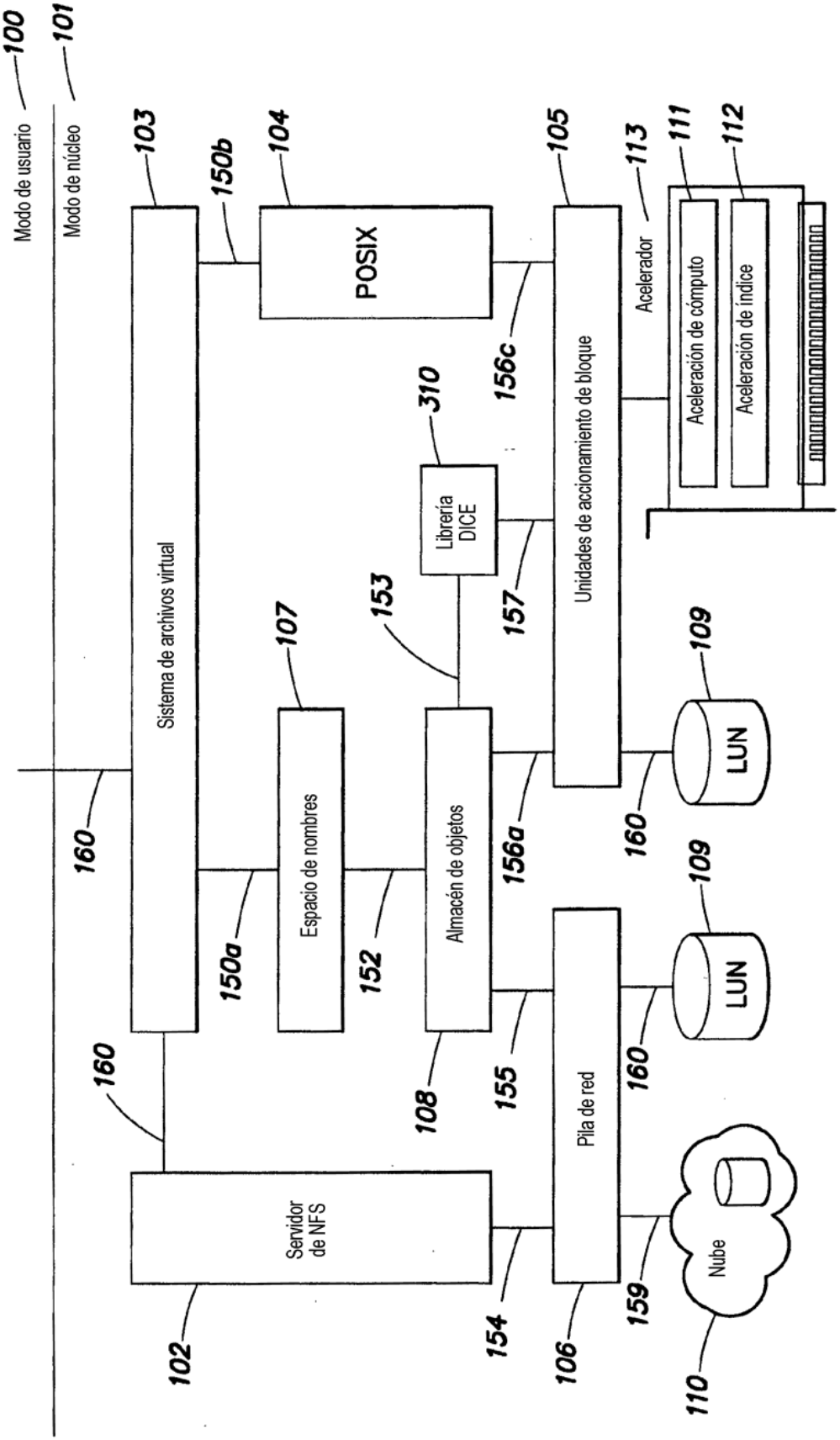
100
101
Modo de usuario
Modo de núcleo
103
150b
104
105
113
111
112
Sistema de archivos virtual
POSIX
Acelerador
Aceleración de cómputo
Aceleración de índice
310
Librería DICE
156c
Unidades de accionamiento de bloque
153
157
107
109
LUN
160
Espacio de nombres
Almacén de objetos
156a
150a
152
108
155
Pila de red
Servidor de NFS
Nube
102
154
106
159
110

FIG. 1

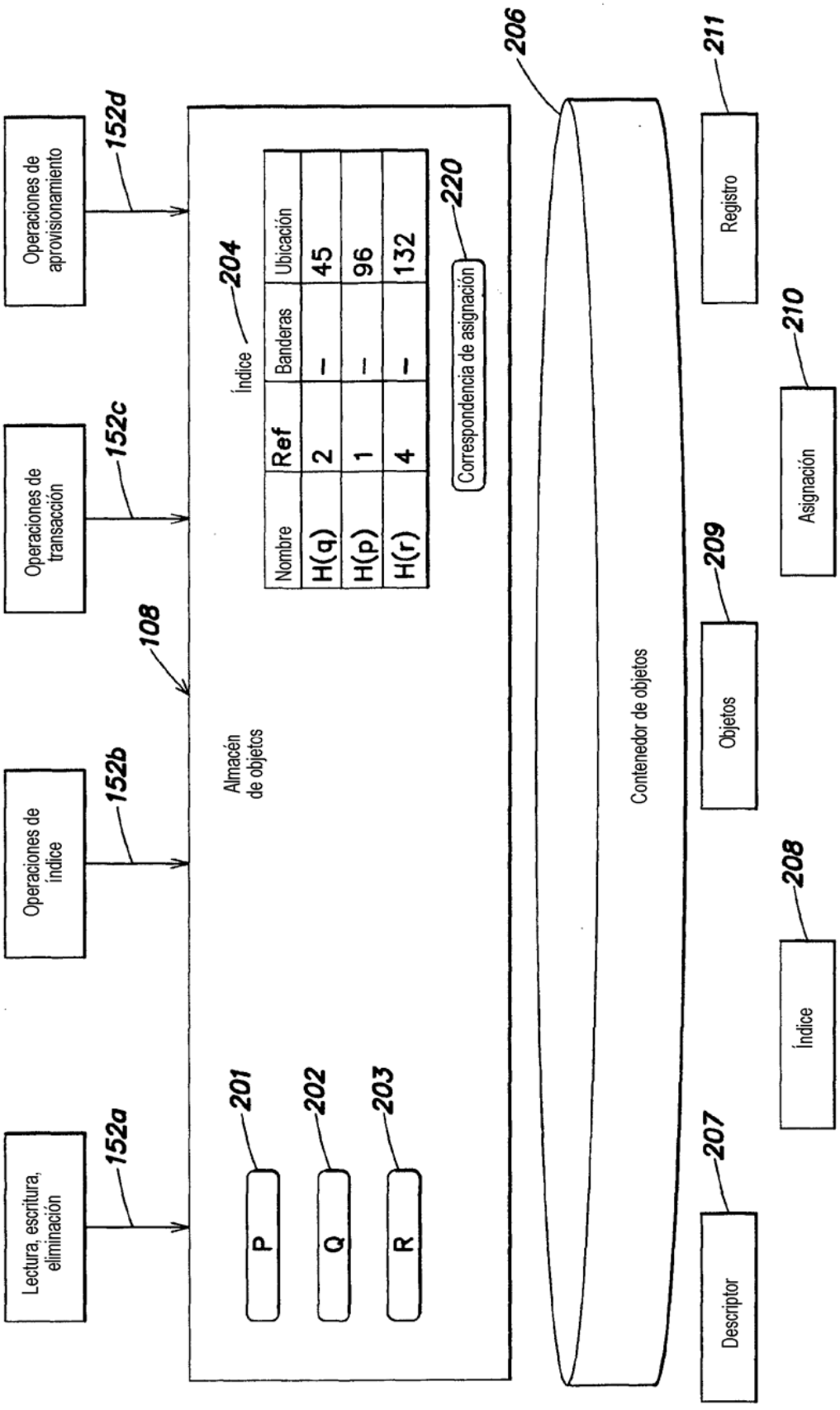
Lectura, escritura, eliminación
152a
Operaciones de índice
152b
Operaciones de transacción
152c
Operaciones de aprovisionamiento
152d
108
Almacén de objetos
201
P
202
Q
203
R
204
Índice
Nombre
Ref
Banderas
Ubicación
H(q)
2
–
45
H(p)
1
–
96
H(r)
4
–
132
220
Correspondencia de asignación
206
Contenedor de objetos
207
Descriptor
208
Índice
209
Objetos
210
Asignación
211
Registro

FIG. 2

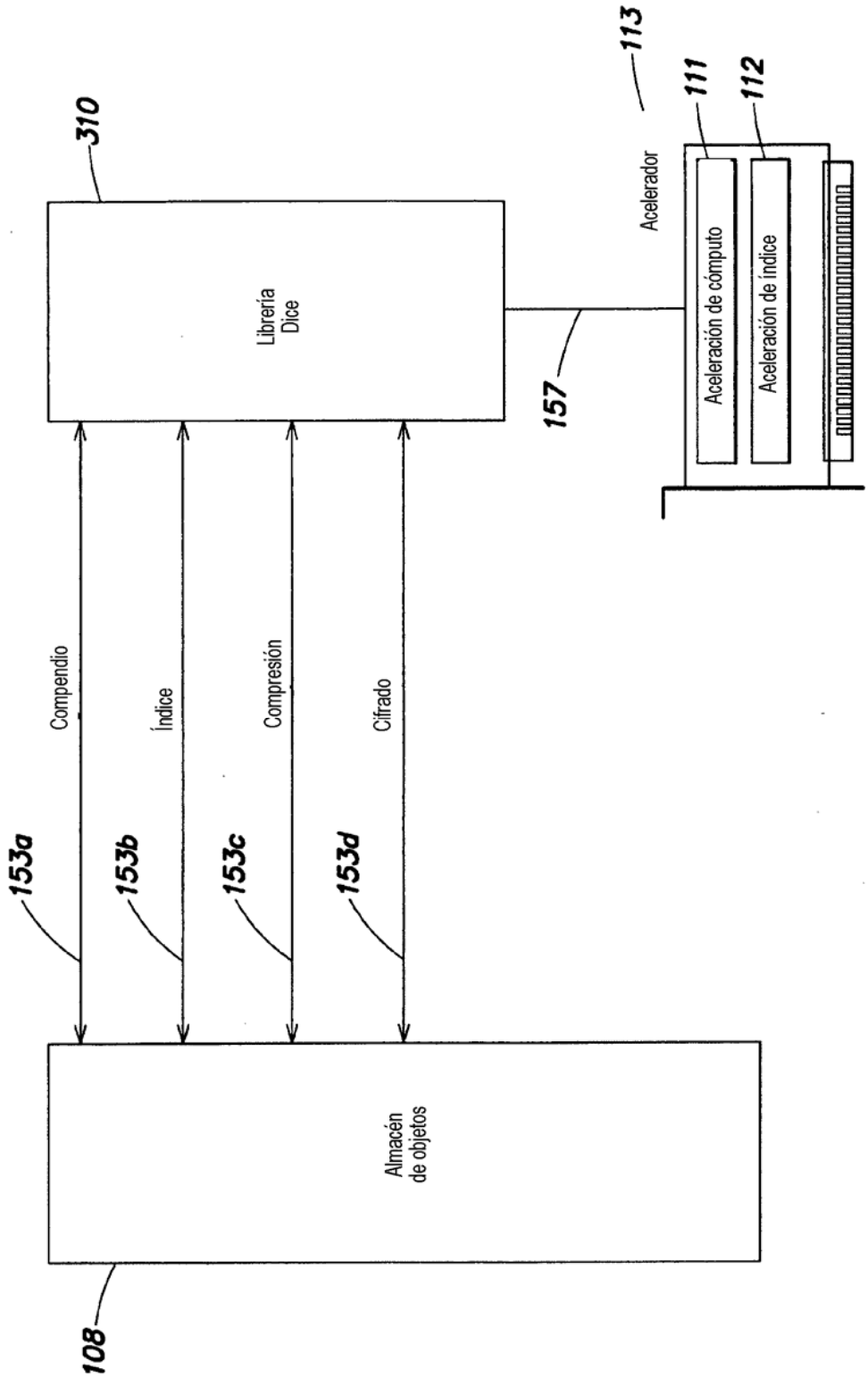
310
Librería
Dice
113
Acelerador
111
Aceleración de cómputo
112
Aceleración de índice
157
Compendio
Índice
Compresión
Cifrado
153a
153b
153c
153d
Almacén
de objetos
108

FIG. 3

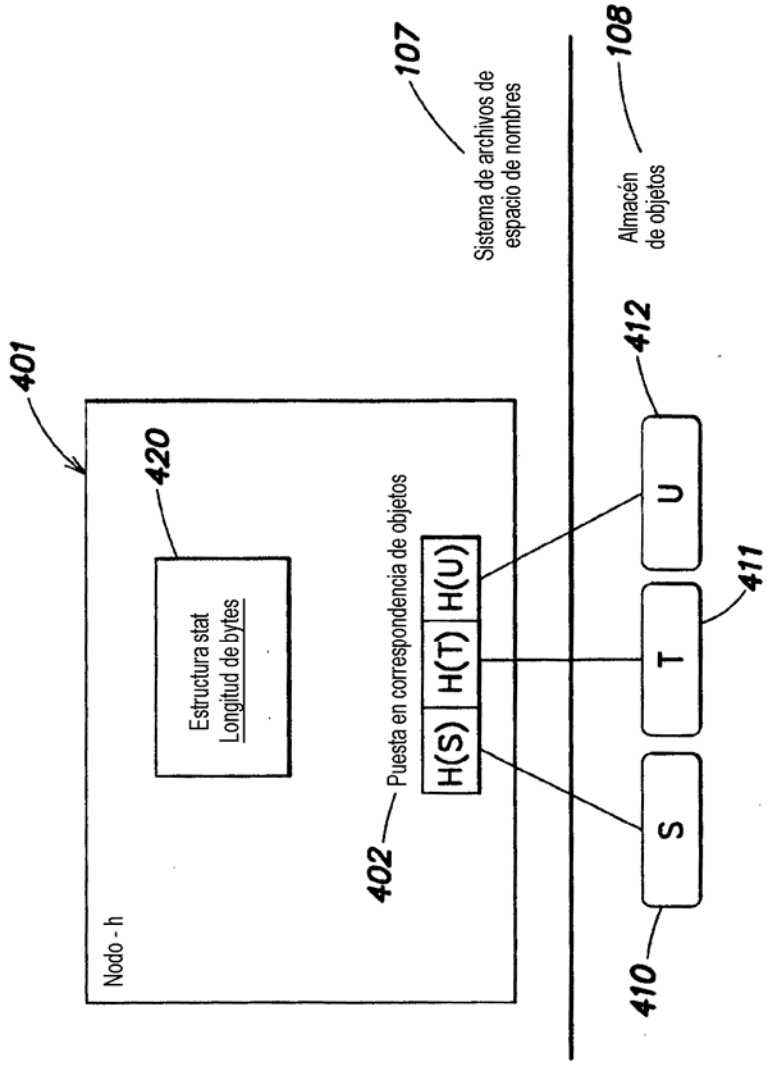
401
420
Estructura stat
Longitud de bytes
Nodo - h
402
Puesta en correspondencia de objetos
H(S)
H(T)
H(U)
107
Sistema de archivos de espacio de nombres
108
Almacén de objetos
412
U
411
T
410
S

FIG. 4

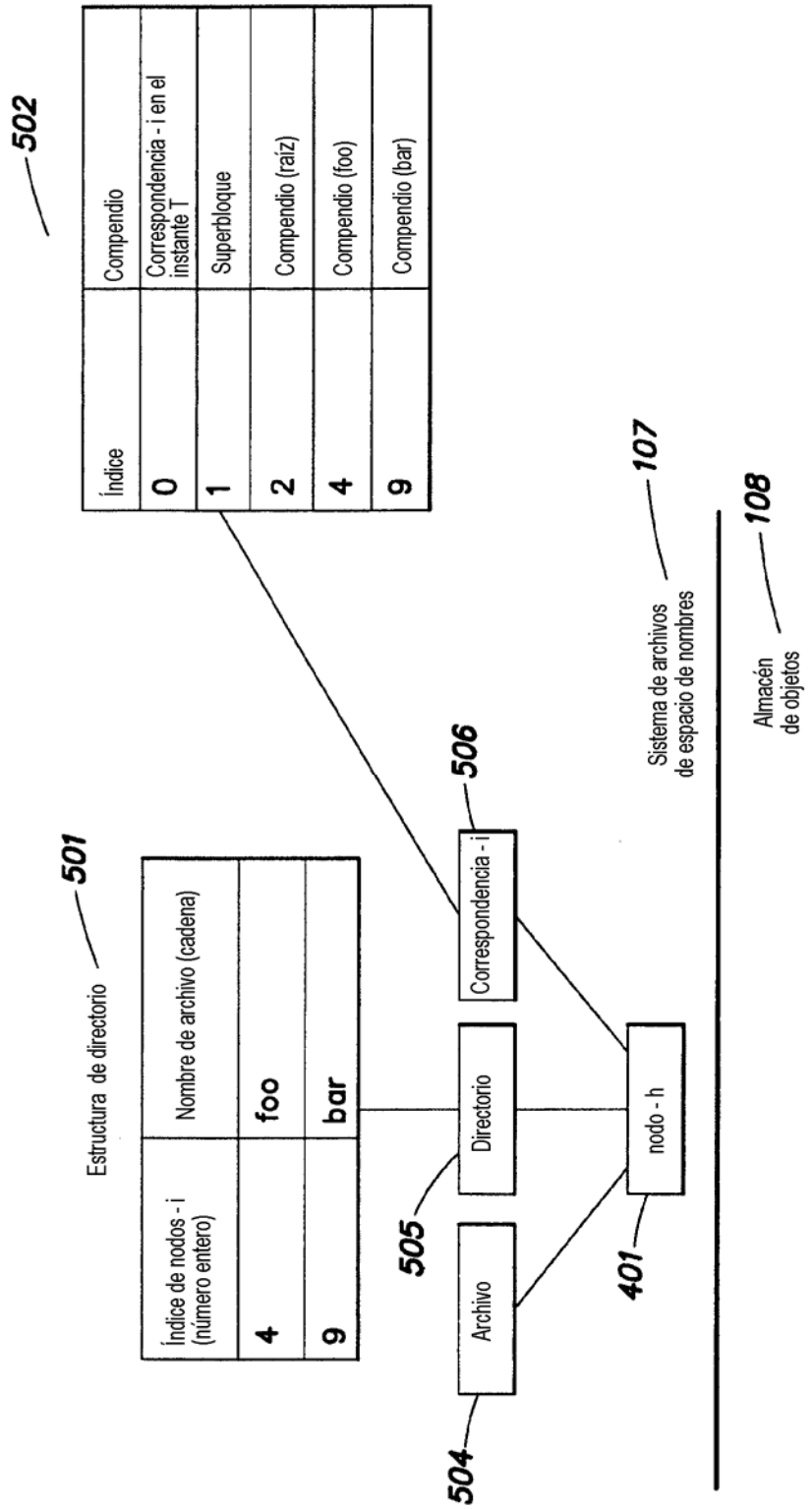
502
Índice
Compendio
0
Correspondencia - i en el instante T
1
Superbloque
2
Compendio (raíz)
4
Compendio (foo)
9
Compendio (bar)
501
Estructura de directorio
Índice de nodos - i (número entero)
Nombre de archivo (cadena)
4
foo
9
bar
506
Correspondencia - i
505
Directorio
504
Archivo
401
nodo - h
107
Sistema de archivos de espacio de nombres
108
Almacén de objetos

FIG. 5

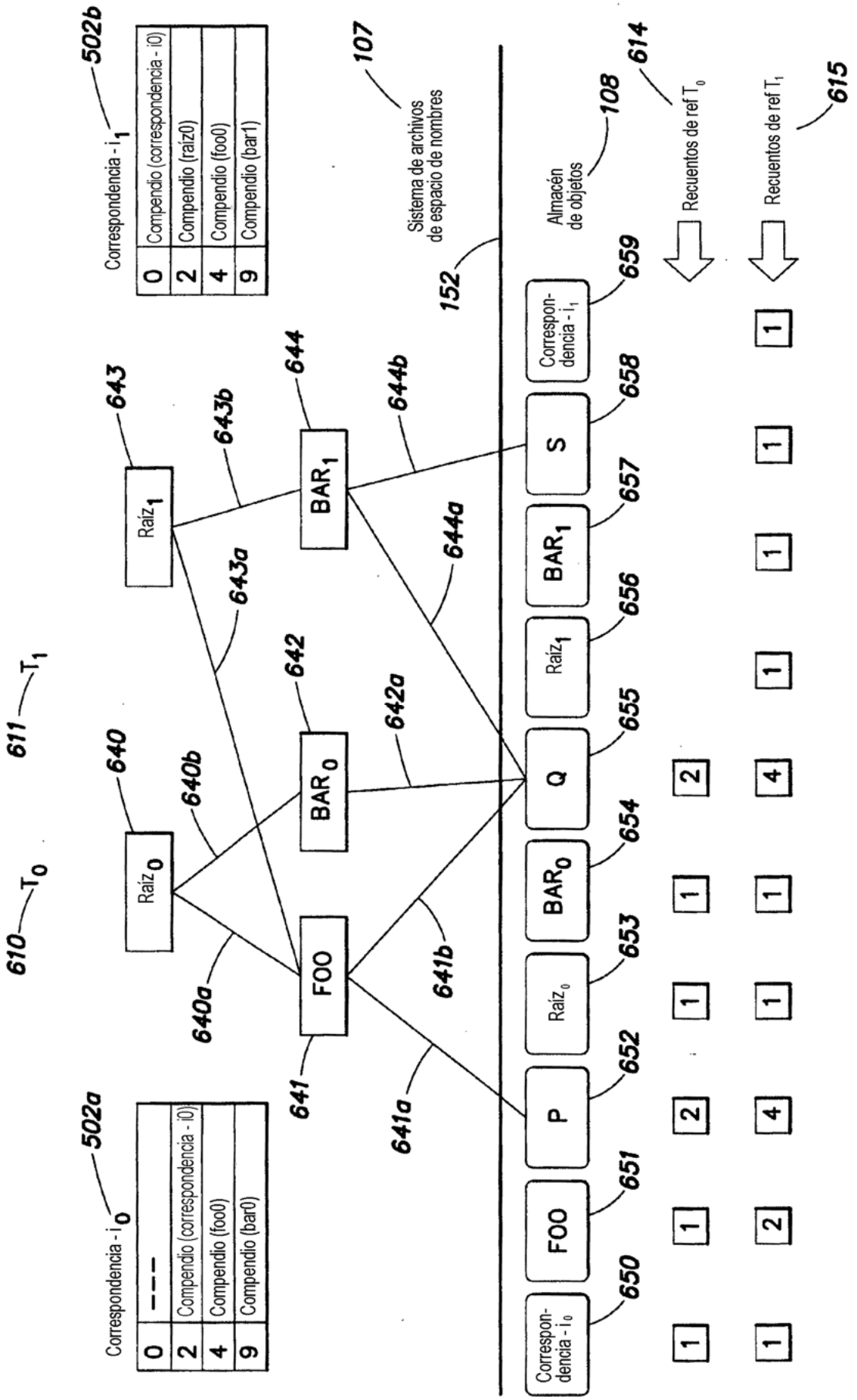
Correspondencia - i_1
0 Compendio (correspondencia - i0)
2 Compendio (raiz0)
4 Compendio (foo0)
9 Compendio (bar1)
Correspondencia - i_0
0 ---
2 Compendio (correspondencia - i0)
4 Compendio (foo0)
9 Compendio (bar0)
Sistema de archivos de espacio de nombres
Almacén de objetos
Recuentos de ref T_0
Recuentos de ref T_1
T_0
T_1
Raíz$_0$
Raíz$_1$
FOO
BAR$_0$
BAR$_1$
P
Q
S
Correspon-dencia - i_0
Correspon-dencia - i_1
610
611
640
640a
640b
641
641a
641b
642
642a
643
643a
643b
644
644a
644b
650
651
652
653
654
655
656
657
658
659
107
108
152
502a
502b
614
615

FIG. 6

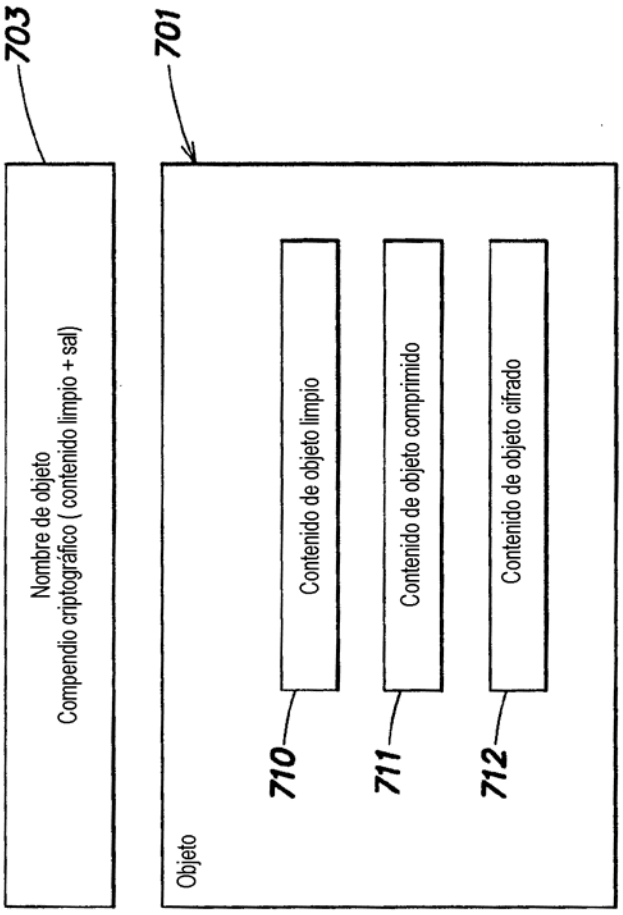
703
701
Nombre de objeto
Compendio criptográfico (contenido limpio + sal)
Objeto
Contenido de objeto limpio
Contenido de objeto comprimido
Contenido de objeto cifrado
710
711
712
El objeto almacenado es 1 de
limpio,
comprimido,
comprimido + cifrado
cifrado

FIG. 7

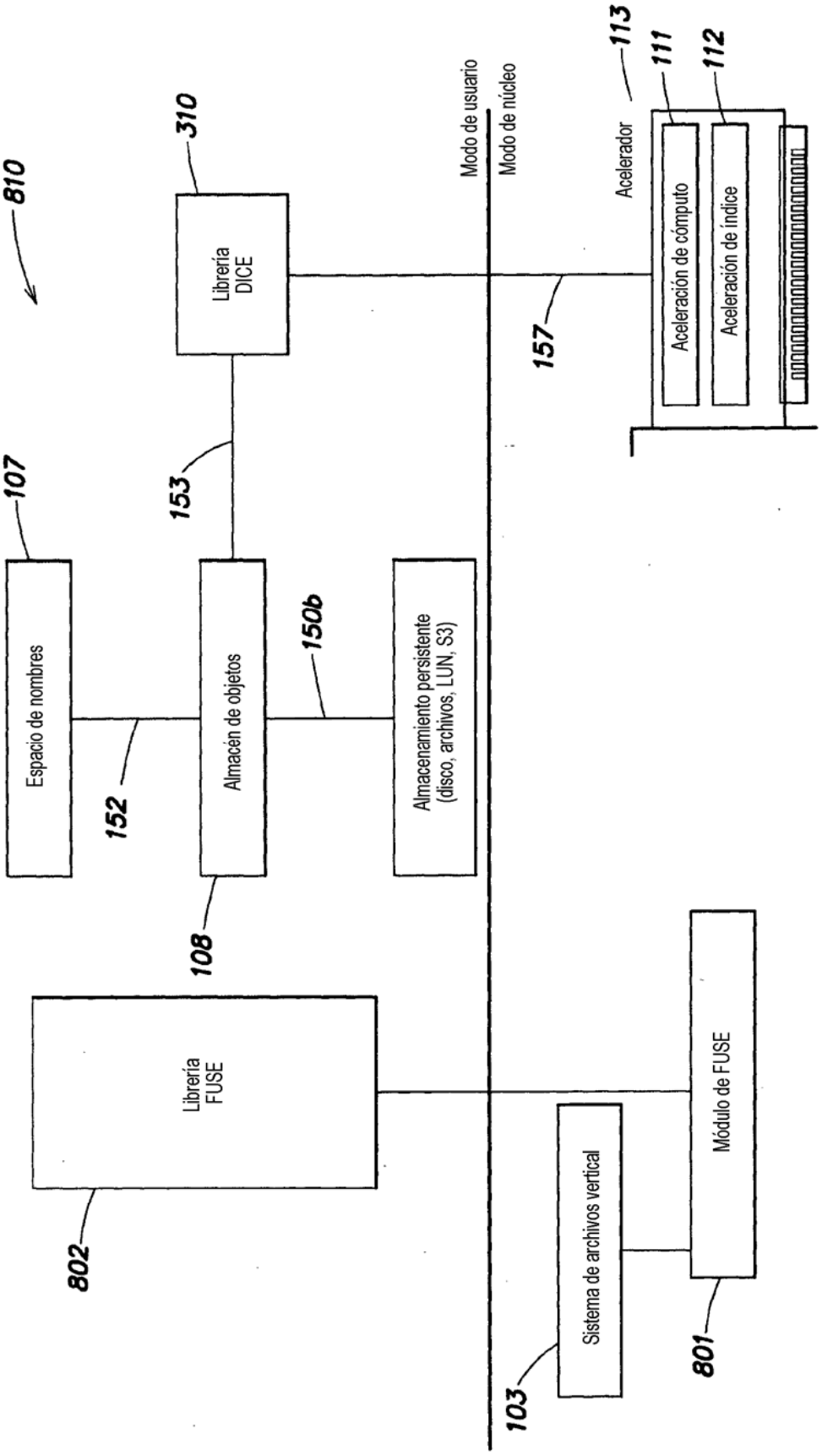
810
310
Librería DICE
Modo de usuario
Modo de núcleo
113
111
112
Acelerador
Aceleración de cómputo
Aceleración de índice
157
153
107
Espacio de nombres
152
Almacén de objetos
150b
Almacenamiento persistente (disco, archivos, LUN, S3)
108
Librería FUSE
802
Sistema de archivos vertical
Módulo de FUSE
801
103

FIG. 8

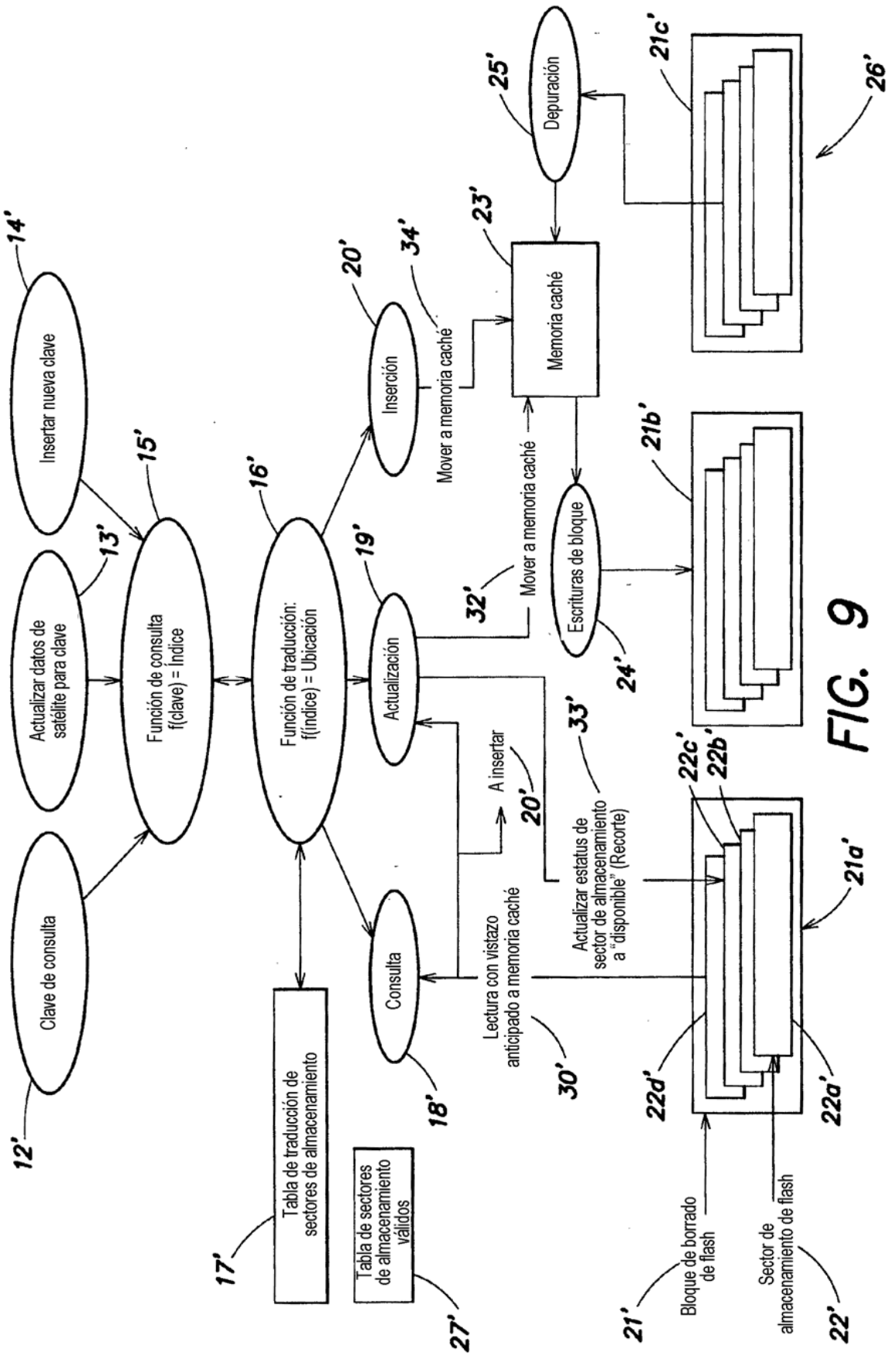
Insertar nueva clave
Actualizar datos de satélite para clave
Clave de consulta
Función de consulta f(clave) = Índice
Función de traducción: f(índice) = Ubicación
Inserción
Actualización
Consulta
Mover a memoria caché
Memoria caché
Depuración
Escrituras de bloque
Mover a memoria caché
A insertar
Lectura con vistazo anticipado a memoria caché
Actualizar estatus de sector de almacenamiento a "disponible" (Recorte)
Tabla de traducción de sectores de almacenamiento
Tabla de sectores de almacenamiento válidos
Bloque de borrado de flash
Sector de almacenamiento de flash
12'
13'
14'
15'
16'
17'
18'
19'
20'
20'
21'
21a'
21b'
21c'
22'
22a'
22b'
22c'
22d'
23'
24'
25'
26'
27'
30'
32'
33'
34'

FIG. 9

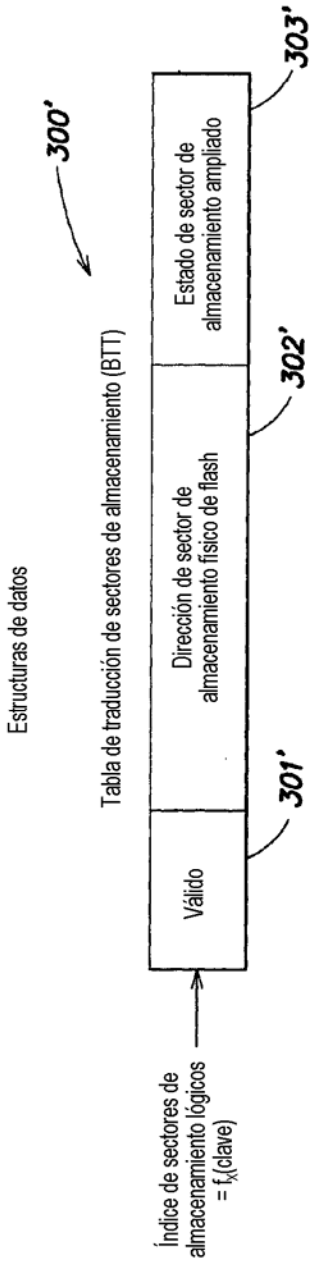

FIG. 10A

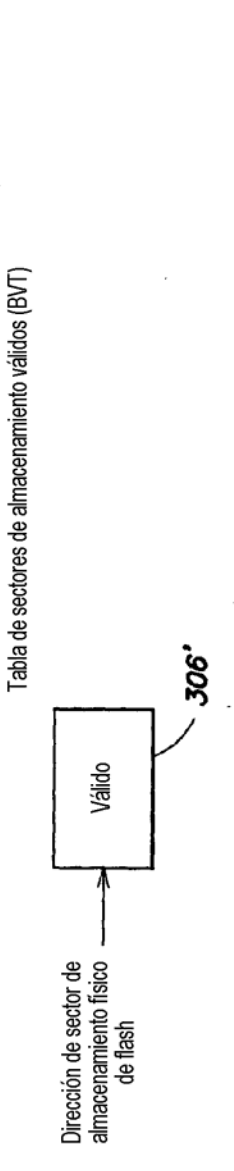

FIG. 10B

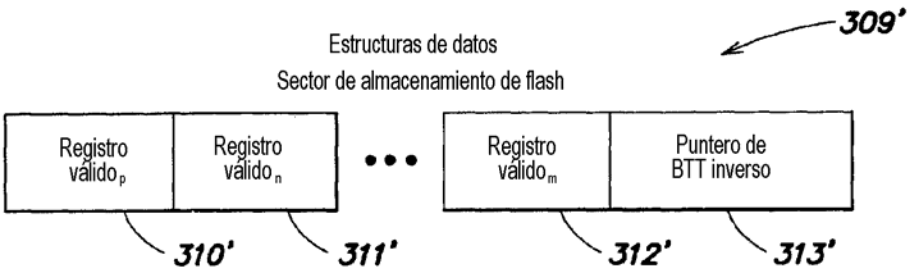

FIG. 10C

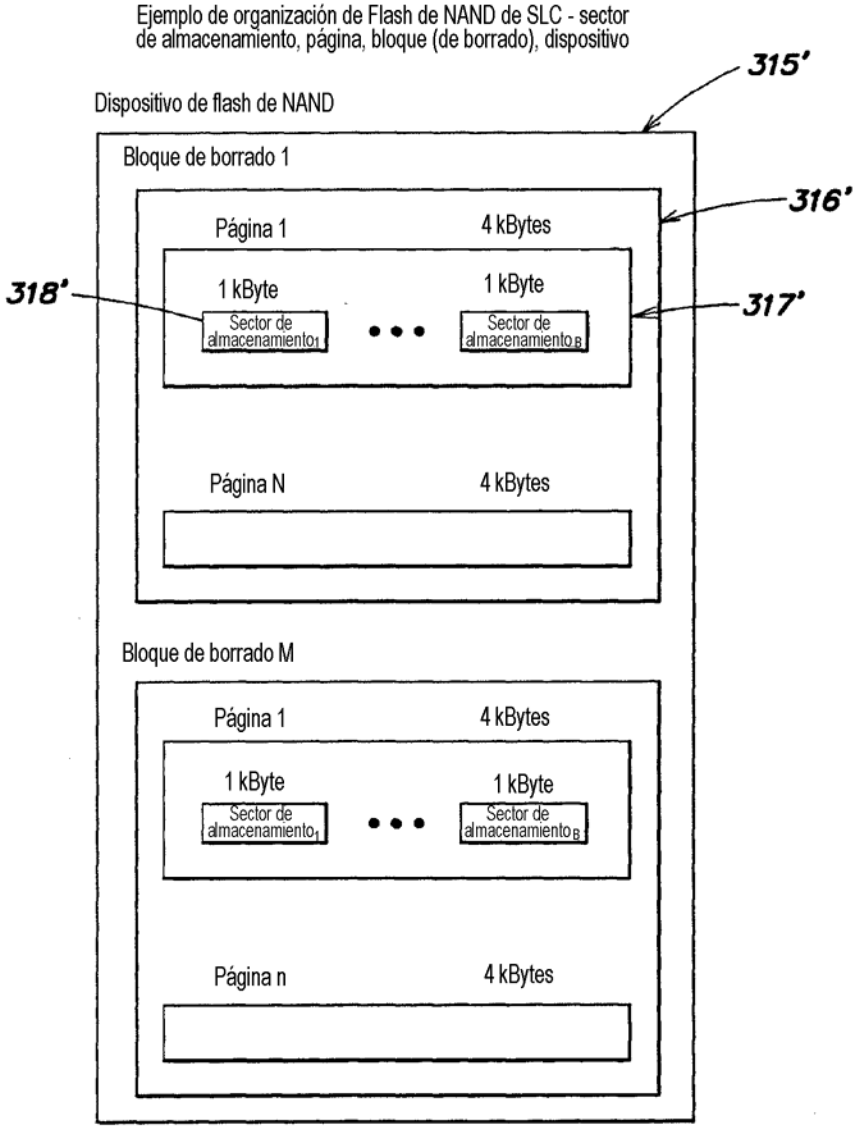

FIG. 10D

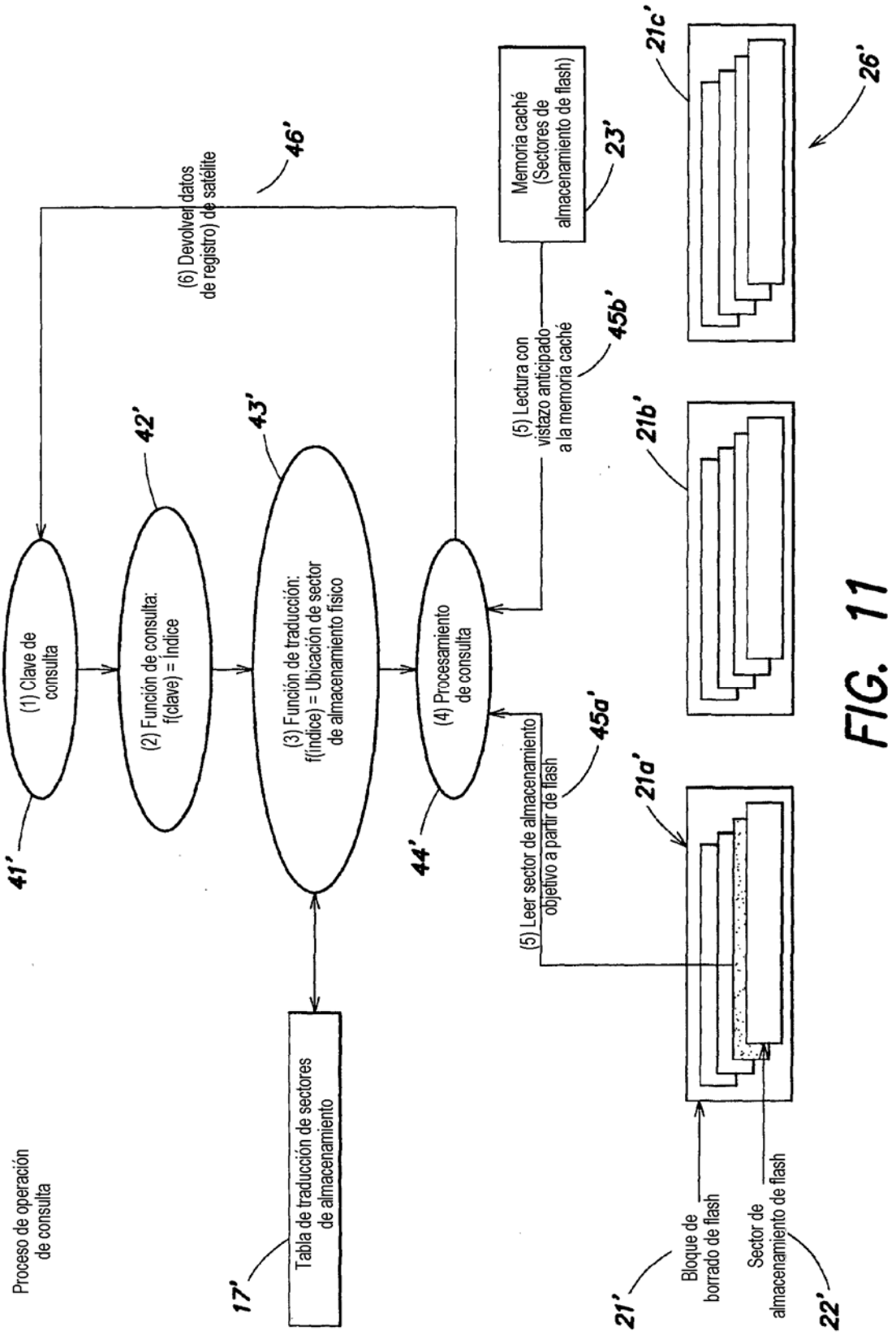
Proceso de operación de consulta
(1) Clave de consulta
(2) Función de consulta: f(clave) = Índice
(3) Función de traducción: f(índice) = Ubicación de sector de almacenamiento físico
(4) Procesamiento de consulta
(5) Leer sector de almacenamiento objetivo a partir de flash
(5) Lectura con vistazo anticipado a la memoria caché
(6) Devolver datos de registro) de satélite
Memoria caché (Sectores de almacenamiento de flash)
Tabla de traducción de sectores de almacenamiento
Bloque de borrado de flash
Sector de almacenamiento de flash
41'
42'
43'
44'
45a'
45b'
46'
17'
21'
21a'
21b'
21c'
22'
23'
26'

FIG. 11

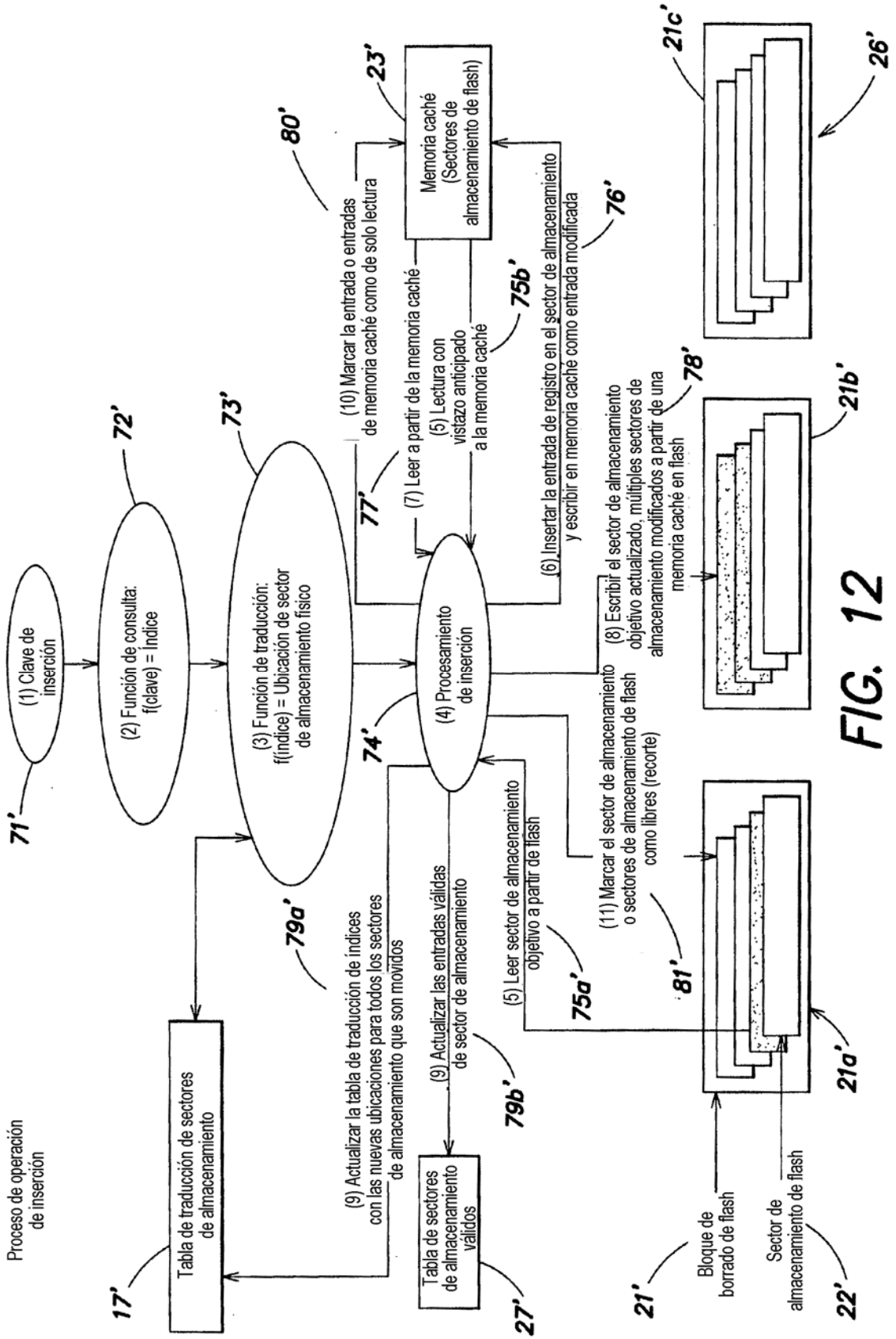
Proceso de operación de inserción
(1) Clave de inserción
(2) Función de consulta: f(clave) = Índice
(3) Función de traducción: f(índice) = Ubicación de sector de almacenamiento físico
(4) Procesamiento de inserción
(5) Leer sector de almacenamiento objetivo a partir de flash
(5) Lectura con vistazo anticipado a la memoria caché
(6) Insertar la entrada de registro en el sector de almacenamiento y escribir en memoria caché como entrada modificada
(7) Leer a partir de la memoria caché
(8) Escribir el sector de almacenamiento objetivo actualizado, múltiples sectores de almacenamiento modificados a partir de una memoria caché en flash
(9) Actualizar la tabla de traducción de índices con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos
(9) Actualizar las entradas válidas de sector de almacenamiento
(10) Marcar la entrada o entradas de memoria caché como de solo lectura
(11) Marcar el sector de almacenamiento o sectores de almacenamiento de flash como libres (recorte)
Memoria caché (Sectores de almacenamiento de flash)
Tabla de traducción de sectores de almacenamiento
Tabla de sectores de almacenamiento válidos
Bloque de borrado de flash
Sector de almacenamiento de flash
71'
72'
73'
74'
75a'
75b'
76'
77'
78'
79a'
79b'
80'
81'
17'
21'
21a'
21b'
21c'
22'
23'
26'
27'

FIG. 12

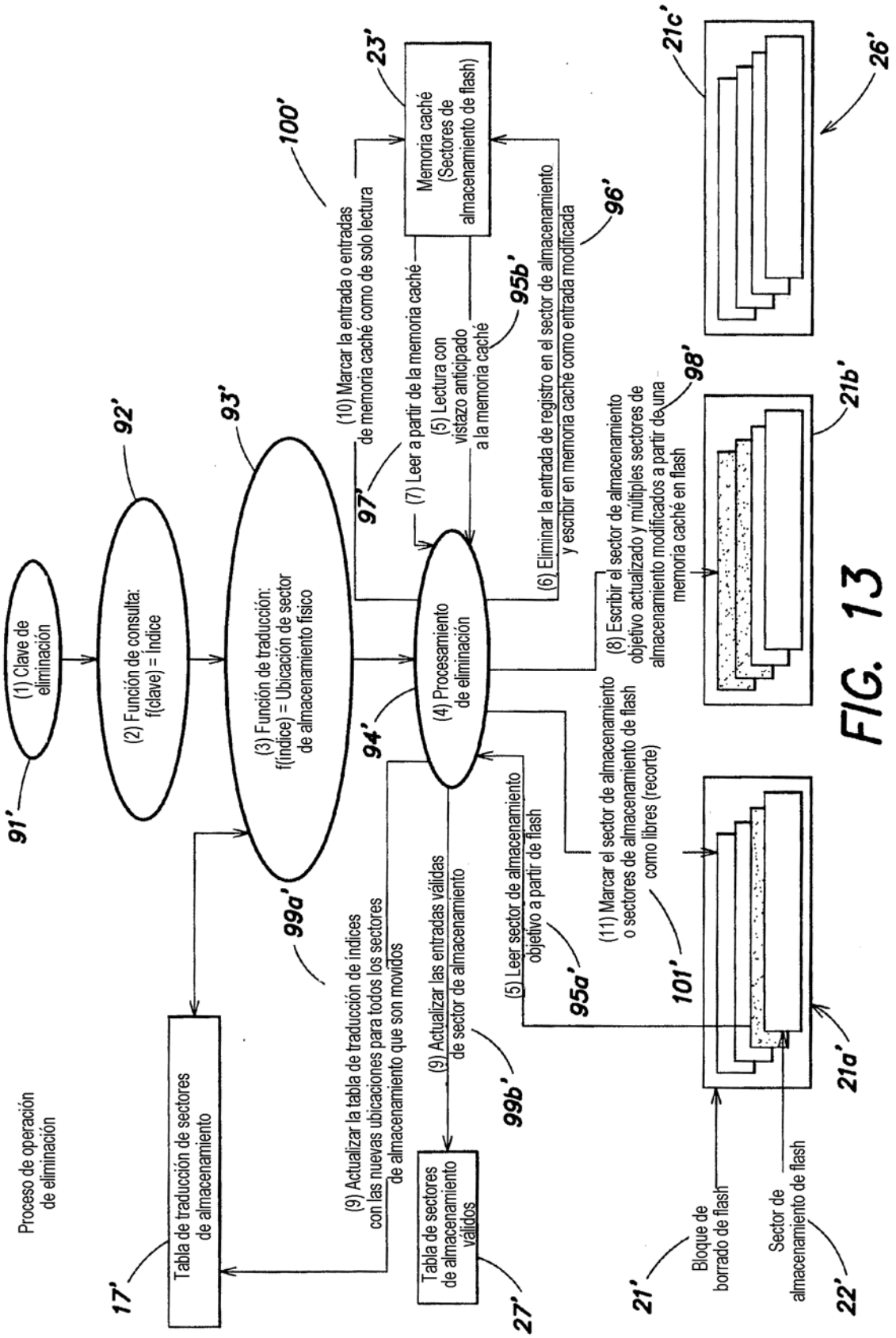
Proceso de operación de eliminación
(1) Clave de eliminación
(2) Función de consulta: f(clave) = Índice
(3) Función de traducción: f(índice) = Ubicación de sector de almacenamiento físico
(4) Procesamiento de eliminación
(5) Leer sector de almacenamiento objetivo a partir de flash
(5) Lectura con vistazo anticipado a la memoria caché
(6) Eliminar la entrada de registro en el sector de almacenamiento y escribir en memoria caché como entrada modificada
(7) Leer a partir de la memoria caché
(8) Escribir el sector de almacenamiento objetivo actualizado y múltiples sectores de almacenamiento modificados a partir de una memoria caché en flash
(9) Actualizar las entradas válidas de sector de almacenamiento
(9) Actualizar la tabla de traducción de índices con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos
(10) Marcar la entrada o entradas de memoria caché como de solo lectura
(11) Marcar el sector de almacenamiento o sectores de almacenamiento de flash como libres (recorte)
Tabla de traducción de sectores de almacenamiento
Tabla de sectores de almacenamiento válidos
Memoria caché (Sectores de almacenamiento de flash)
Bloque de borrado de flash
Sector de almacenamiento de flash
91'
92'
93'
94'
95a'
95b'
96'
97'
98'
99a'
99b'
100'
101'
17'
21'
21a'
21b'
21c'
22'
23'
26'
27'

FIG. 13

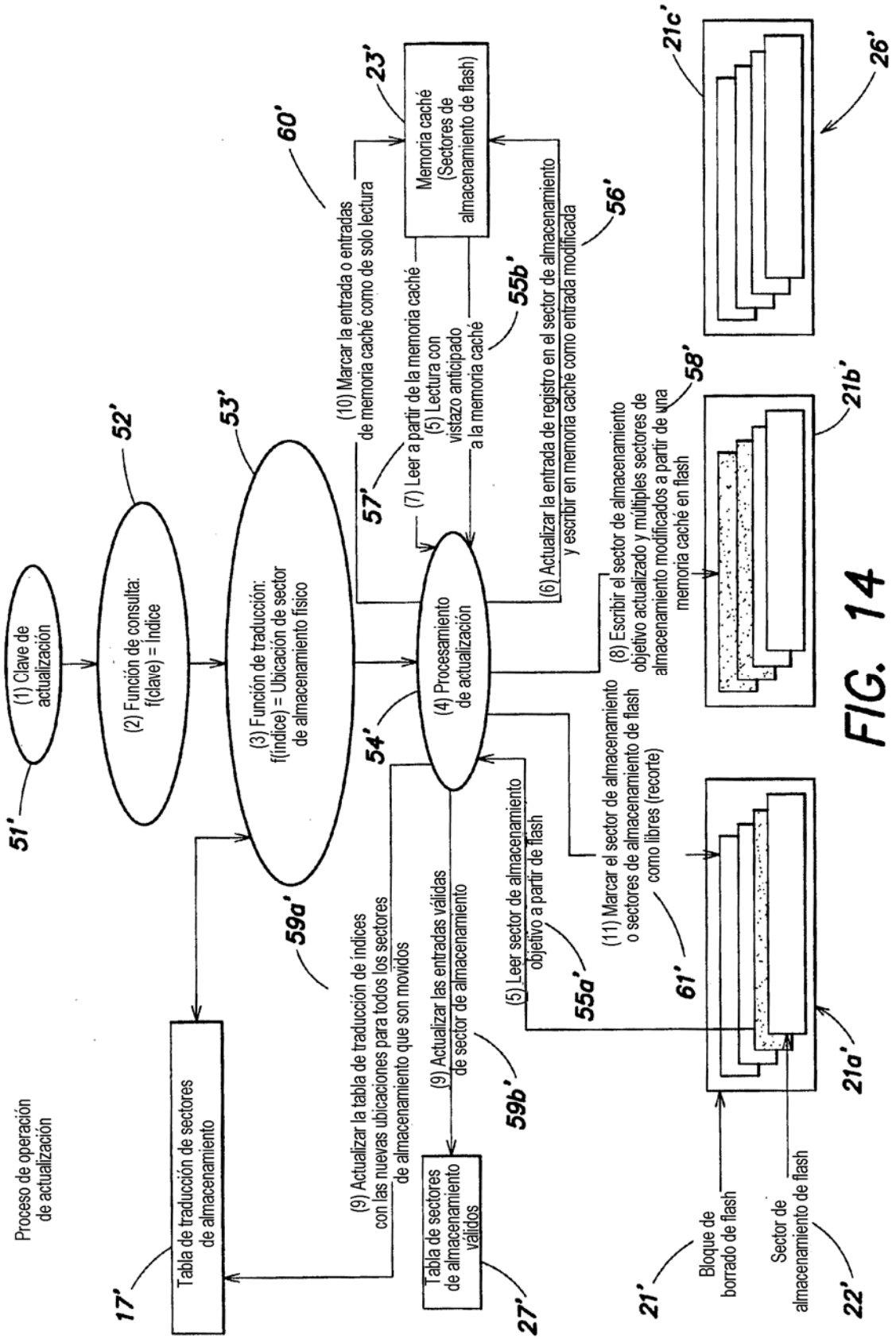
Proceso de operación de actualización
(1) Clave de actualización
(2) Función de consulta: f(clave) = índice
(3) Función de traducción: f(índice) = Ubicación de sector de almacenamiento físico
(4) Procesamiento de actualización
(5) Leer sector de almacenamiento objetivo a partir de flash
(5) Lectura con vistazo anticipado a la memoria caché
(6) Actualizar la entrada de registro en el sector de almacenamiento y escribir en memoria caché como entrada modificada
(7) Leer a partir de la memoria caché
(8) Escribir el sector de almacenamiento objetivo actualizado y múltiples sectores de almacenamiento modificados a partir de una memoria caché en flash
(9) Actualizar las entradas válidas de sector de almacenamiento
(9) Actualizar la tabla de traducción de índices con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos
(10) Marcar la entrada o entradas de memoria caché como de solo lectura
(11) Marcar el sector de almacenamiento o sectores de almacenamiento de flash como libres (recorte)
Memoria caché (Sectores de almacenamiento de flash)
Tabla de traducción de sectores de almacenamiento
Tabla de sectores de almacenamiento válidos
Bloque de borrado de flash
Sector de almacenamiento de flash
17'
21'
21a'
21b'
21c'
22'
23'
26'
27'
51'
52'
53'
54'
55a'
55b'
56'
57'
58'
59a'
59b'
60'
61'

FIG. 14

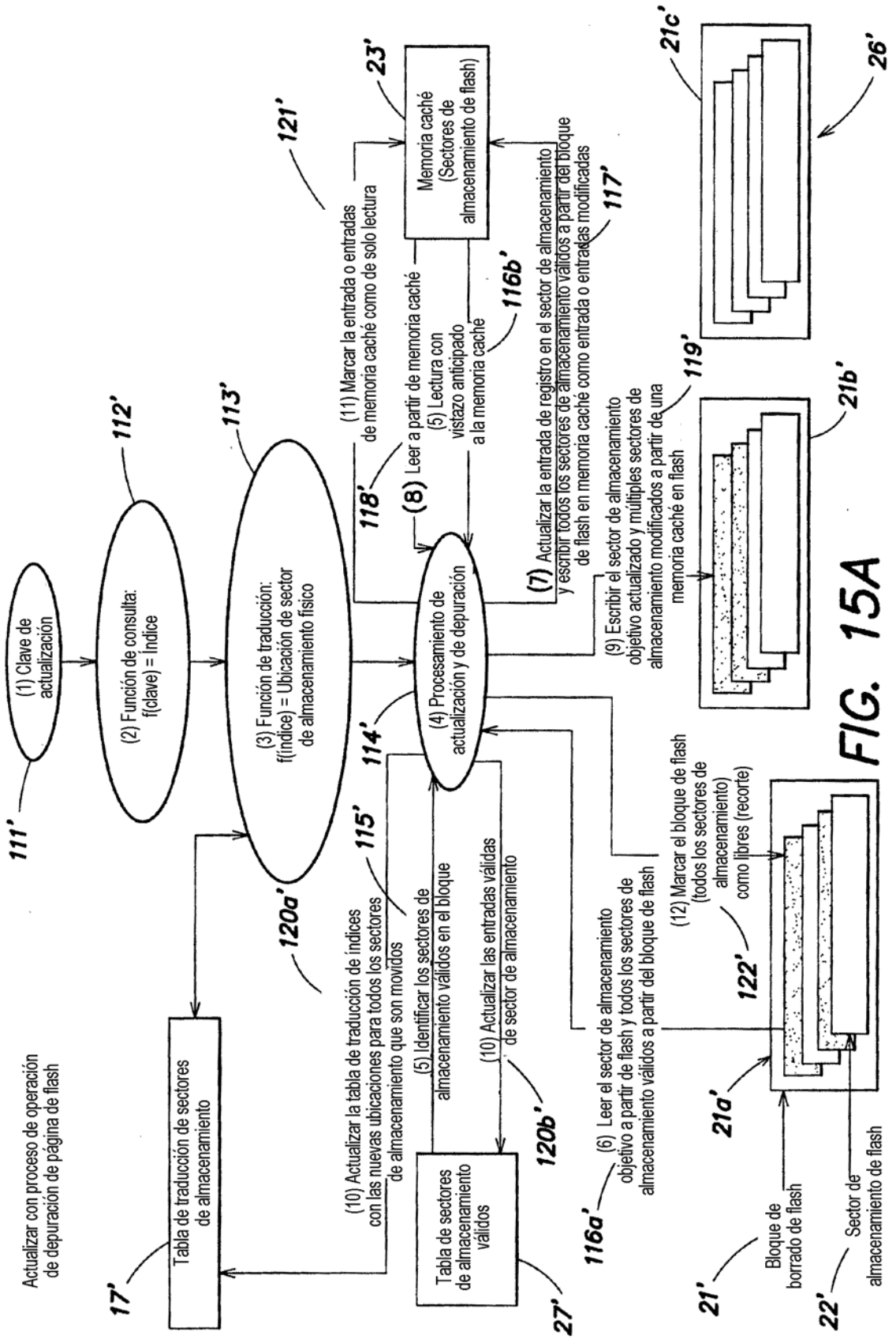
Actualizar con proceso de operación de depuración de página de flash
(1) Clave de actualización
(2) Función de consulta: f(clave) = Índice
(3) Función de traducción: f(índice) = Ubicación de sector de almacenamiento físico
(4) Procesamiento de actualización y de depuración
(5) Identificar los sectores de almacenamiento válidos en el bloque
(6) Leer el sector de almacenamiento objetivo a partir de flash y todos los sectores de almacenamiento válidos a partir del bloque de flash
(7) Actualizar la entrada de registro en el sector de almacenamiento y escribir todos los sectores de almacenamiento válidos a partir del bloque de flash en memoria caché como entrada o entradas modificadas
(8) Leer a partir de memoria caché (5) Lectura con vistazo anticipado a la memoria caché
(9) Escribir el sector de almacenamiento objetivo actualizado y múltiples sectores de almacenamiento modificados a partir de una memoria caché en flash
(10) Actualizar las entradas válidas de sector de almacenamiento
(10) Actualizar la tabla de traducción de índices con las nuevas ubicaciones para todos los sectores de almacenamiento que son movidos
(11) Marcar la entrada o entradas de memoria caché como de solo lectura
(12) Marcar el bloque de flash (todos los sectores de almacenamiento) como libres (recorte)
Tabla de traducción de sectores de almacenamiento
Tabla de sectores de almacenamiento válidos
Memoria caché (Sectores de almacenamiento de flash)
Bloque de borrado de flash
Sector de almacenamiento de flash
111'
112'
113'
114'
115'
116a'
116b'
117'
118'
119'
120a'
120b'
121'
122'
17'
21'
21a'
21b'
21c'
22'
23'
26'
27'

FIG. 15A

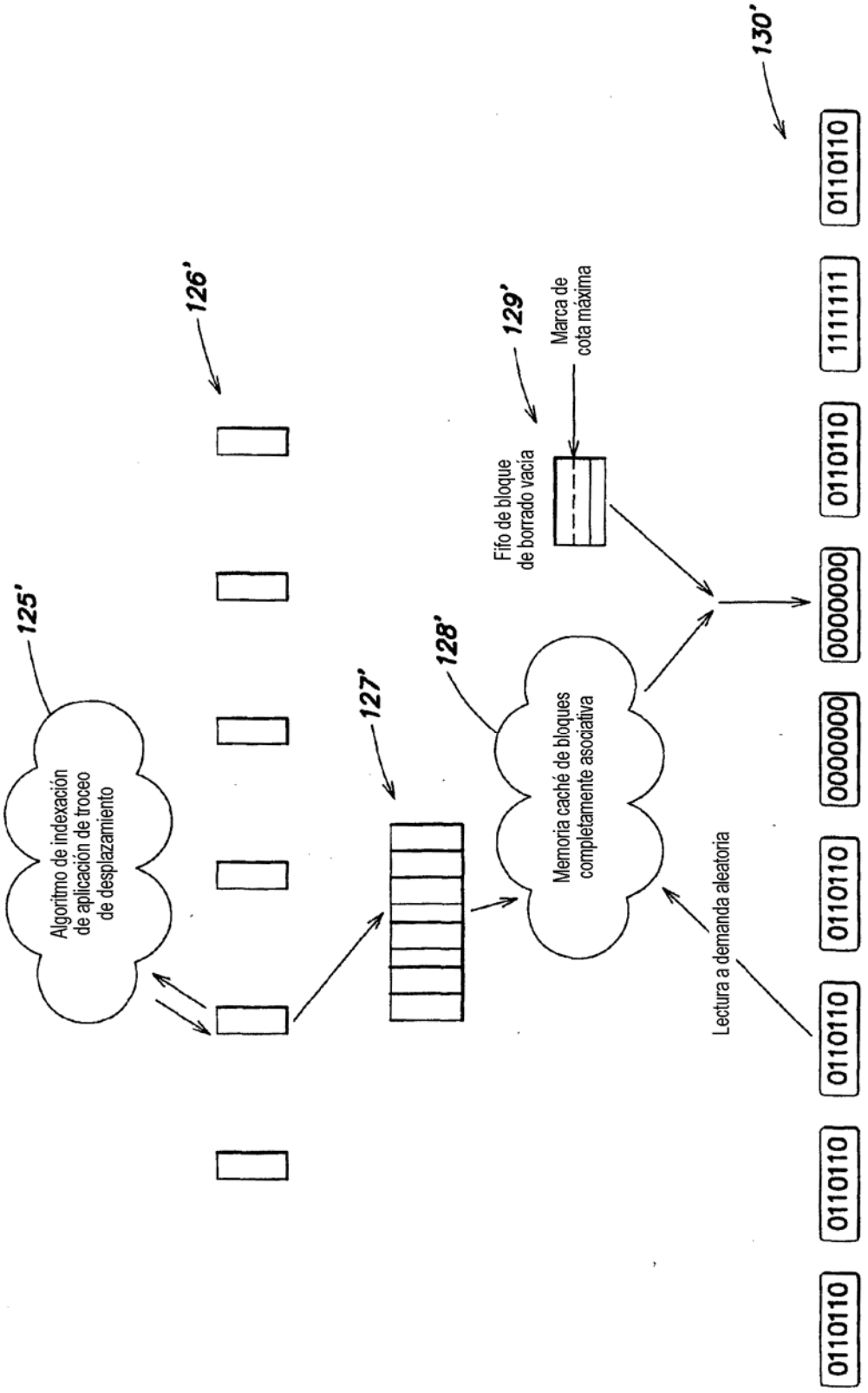
125'
Algoritmo de indexación de aplicación de troceo de desplazamiento
126'
127'
128'
Memoria caché de bloques completamente asociativa
129'
Fifo de bloque de borrado vacía
Marca de cota máxima
130'
Lectura a demanda aleatoria
0110110
0110110
0110110
0110110
0000000
0000000
0110110
1111111
0110110

FIG. 15B

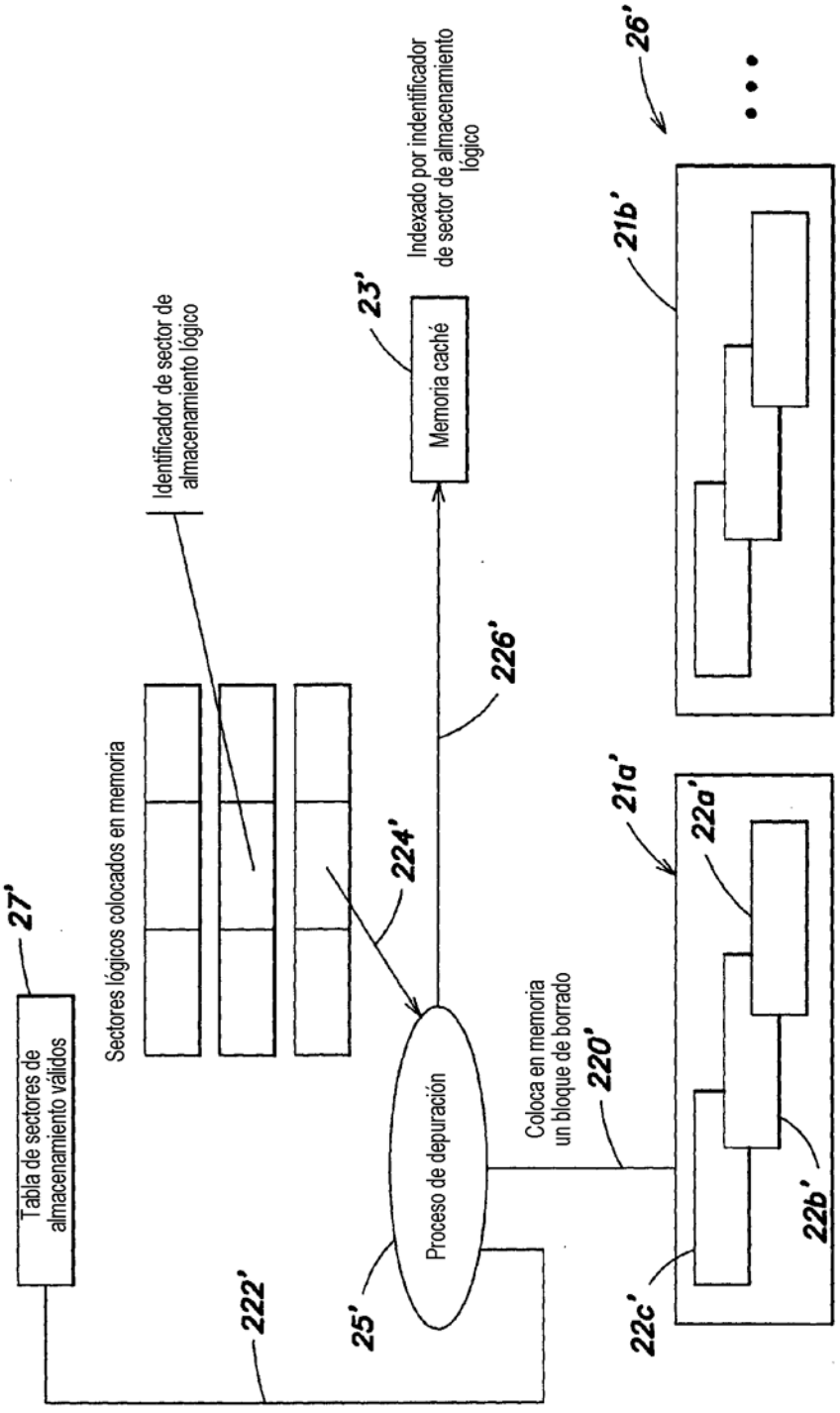
27'
Tabla de sectores de almacenamiento válidos
Sectores lógicos colocados en memoria
Identificador de sector de almacenamiento lógico
224'
226'
23'
Memoria caché
Indexado por indentificador de sector de almacenamiento lógico
26'
21b'
21a'
22a'
22b'
22c'
Coloca en memoria un bloque de borrado
220'
Proceso de depuración
25'
222'

FIG. 16A

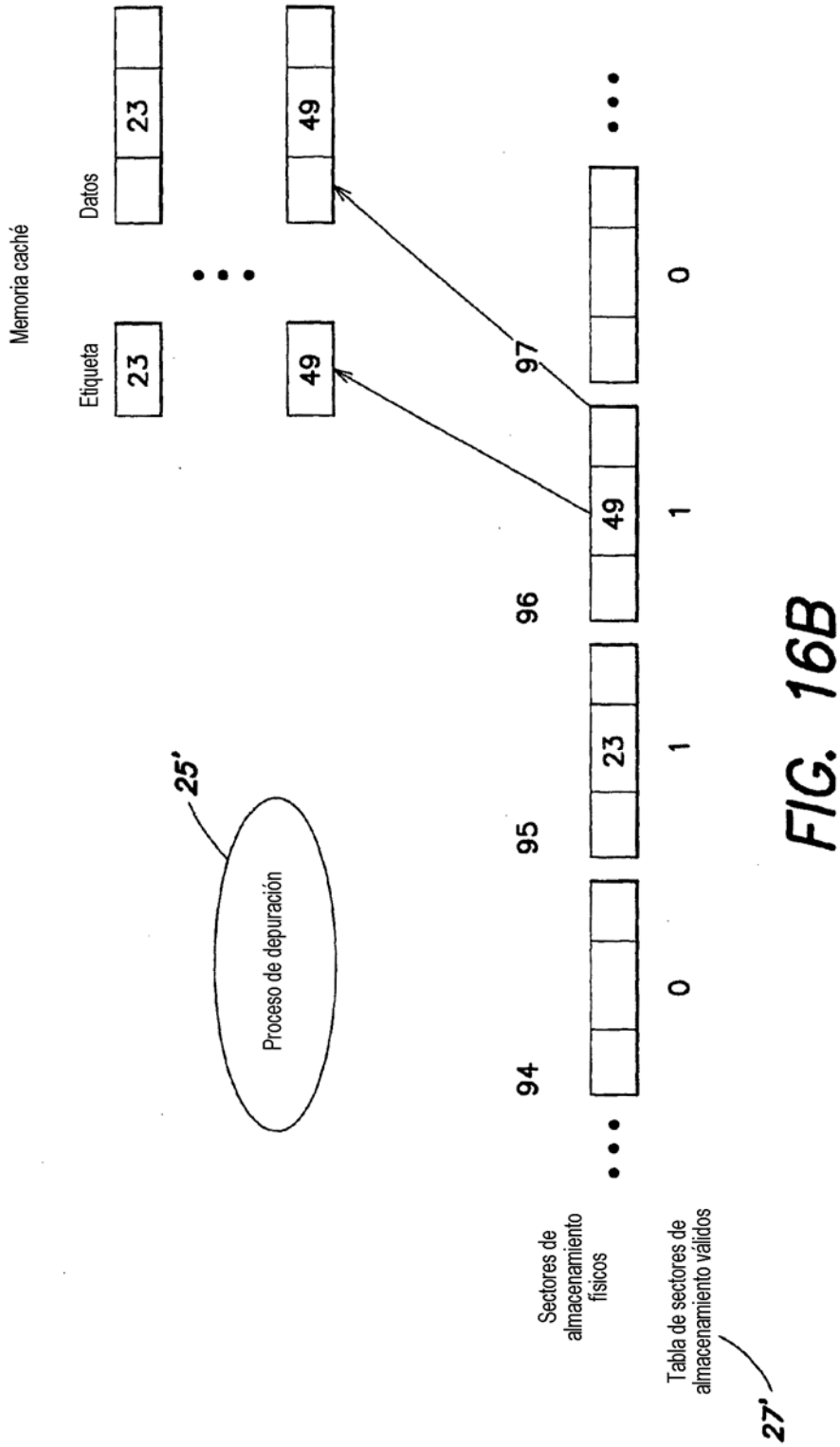
Memoria caché
Etiqueta
Datos
23
49
25'
Proceso de depuración
94
95
96
97
0
1
1
0
Sectores de almacenamiento físicos
Tabla de sectores de almacenamiento válidos
27'

FIG. 16B

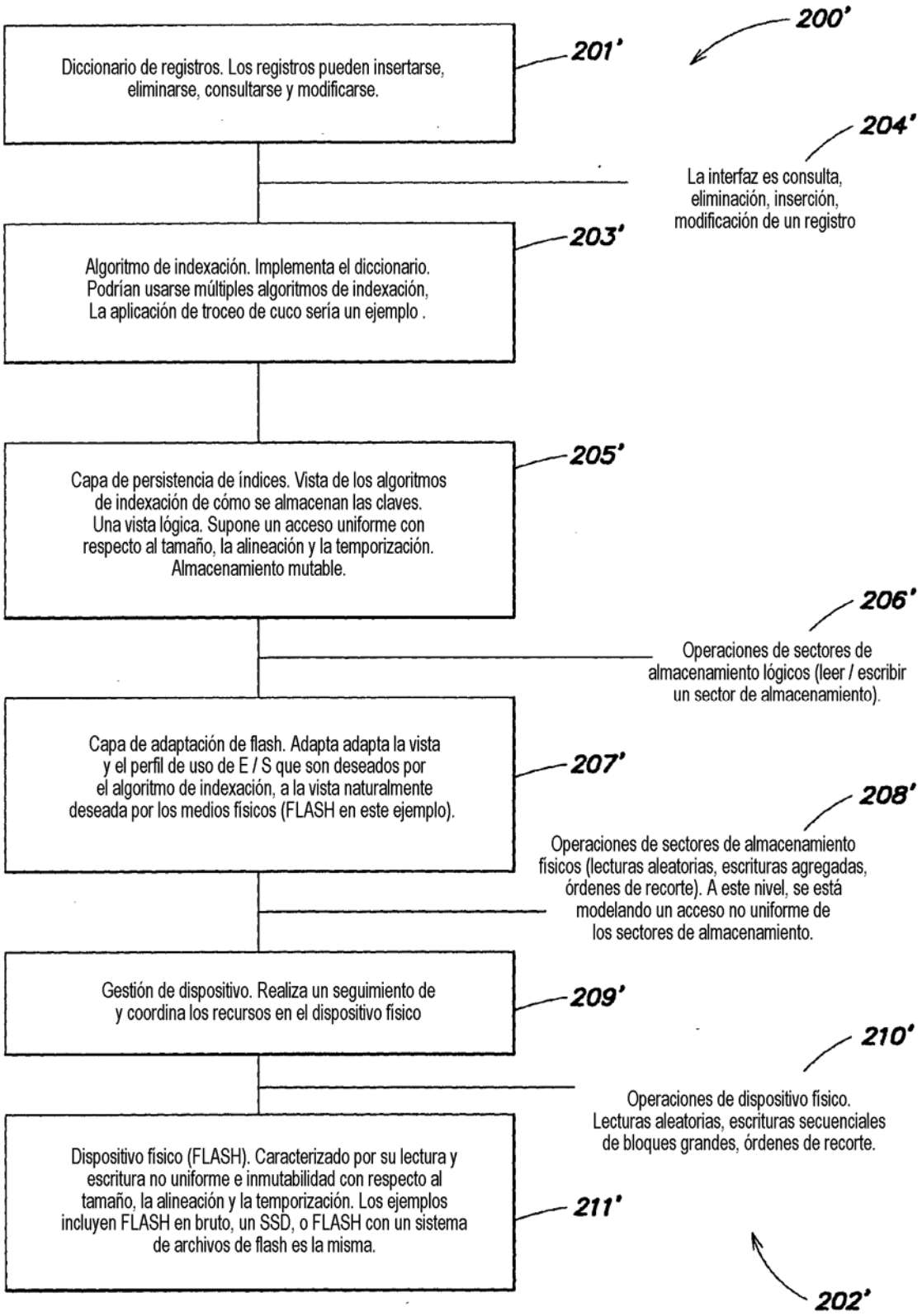
200'
Diccionario de registros. Los registros pueden insertarse, eliminarse, consultarse y modificarse.
201'
204'
La interfaz es consulta, eliminación, inserción, modificación de un registro
Algoritmo de indexación. Implementa el diccionario. Podrían usarse múltiples algoritmos de indexación, La aplicación de troceo de cuco sería un ejemplo .
203'
Capa de persistencia de índices. Vista de los algoritmos de indexación de cómo se almacenan las claves. Una vista lógica. Supone un acceso uniforme con respecto al tamaño, la alineación y la temporización. Almacenamiento mutable.
205'
206'
Operaciones de sectores de almacenamiento lógicos (leer / escribir un sector de almacenamiento).
Capa de adaptación de flash. Adapta adapta la vista y el perfil de uso de E / S que son deseados por el algoritmo de indexación, a la vista naturalmente deseada por los medios físicos (FLASH en este ejemplo).
207'
208'
Operaciones de sectores de almacenamiento físicos (lecturas aleatorias, escrituras agregadas, órdenes de recorte). A este nivel, se está modelando un acceso no uniforme de los sectores de almacenamiento.
Gestión de dispositivo. Realiza un seguimiento de y coordina los recursos en el dispositivo físico
209'
210'
Operaciones de dispositivo físico. Lecturas aleatorias, escrituras secuenciales de bloques grandes, órdenes de recorte.
Dispositivo físico (FLASH). Caracterizado por su lectura y escritura no uniforme e inmutabilidad con respecto al tamaño, la alineación y la temporización. Los ejemplos incluyen FLASH en bruto, un SSD, o FLASH con un sistema de archivos de flash es la misma.
211'
202'

FIG. 17

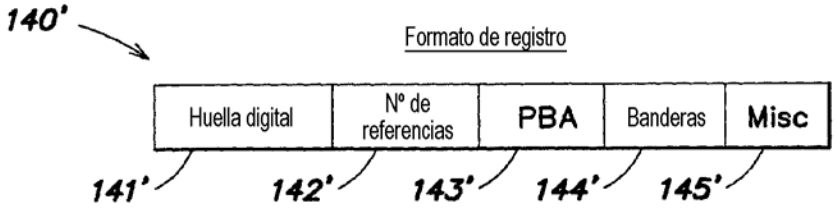

FIG. 18

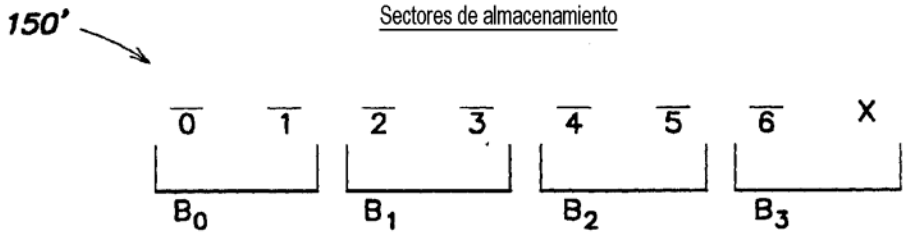

FIG. 20

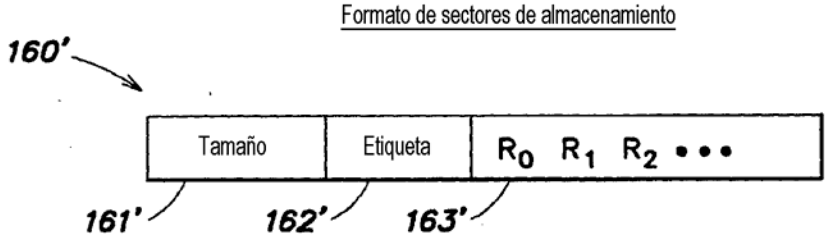

FIG. 21

Aplicación de troceo de desplazamiento (aplicación de troceo de cuco)

	$H_0(X)$	$H_1(X)$
P	2	5
Q	1	3

FIG. 19A

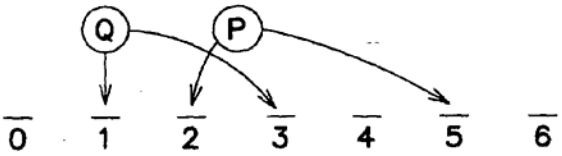

FIG. 19B

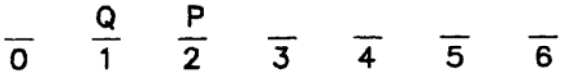

FIG. 19C

	$H_0(R)$	$H_1(R)$
R	1	2

FIG. 19D

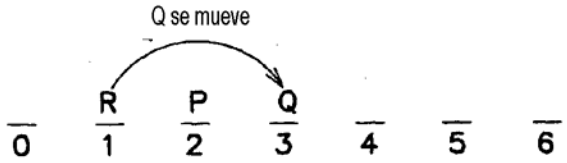

FIG. 19E

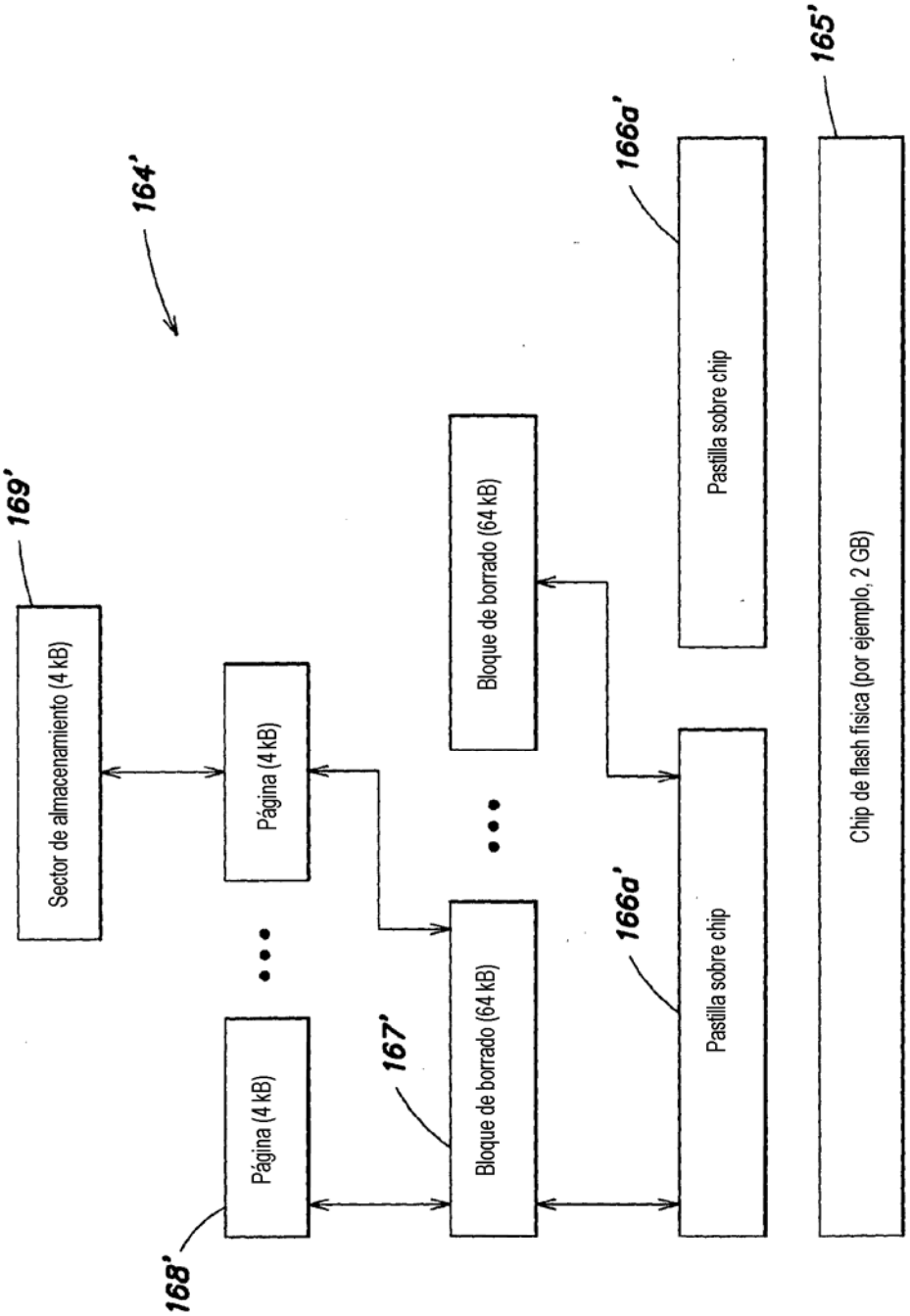
164'
169'
Sector de almacenamiento (4 kB)
Página (4 kB)
Página (4 kB)
168'
167'
Bloque de borrado (64 kB)
Bloque de borrado (64 kB)
166a'
Pastilla sobre chip
Pastilla sobre chip
165'
Chip de flash física (por ejemplo, 2 GB)

FIG. 22

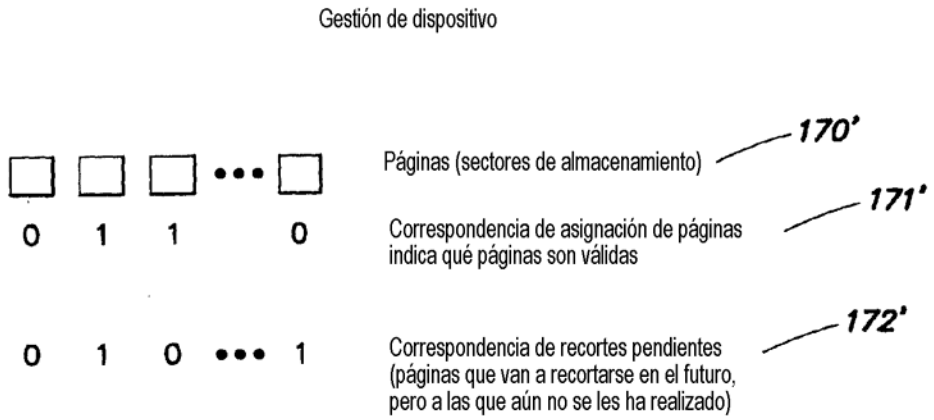

FIG. 23A

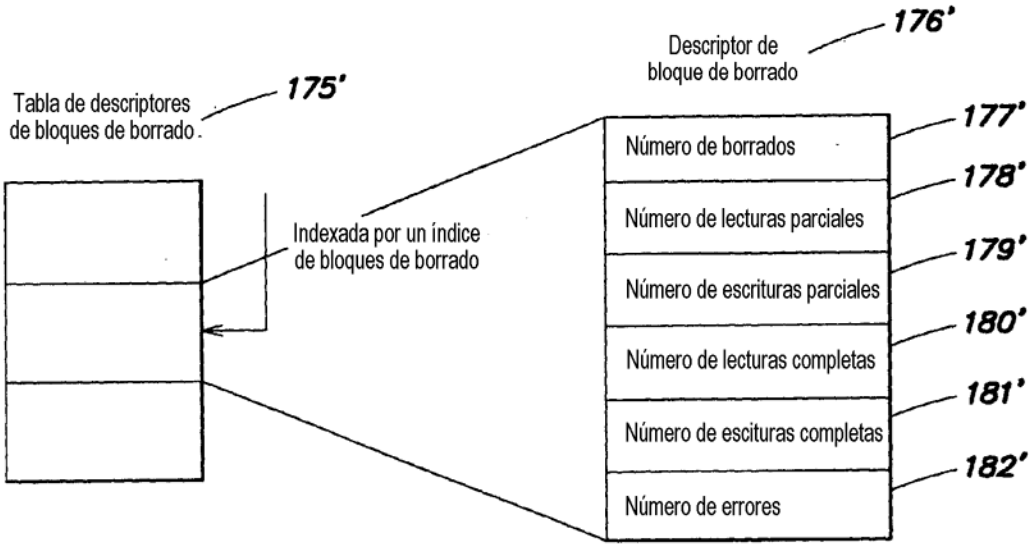

FIG. 23B