(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 555 275**

(51) Int. CI.:

G06F 9/50 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **15.06.2012 E 12732904 (3)**

(97) Fecha y número de publicación de la concesión europea: **23.09.2015 EP 2721489**

(54) Título: **Máquina virtual de software para aceleración de procesamiento de datos transaccionales**

(30) Prioridad:

16.06.2011 US 201161497860 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
30.12.2015

(73) Titular/es:

ARGYLE DATA, INC. (100.0%)
2755 Campus Drive, Suite 165
San Mateo CA 94403, US

(72) Inventor/es:

HUETTER, RAYMOND J. y
YAMARTI, ALKA

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Máquina virtual de software para aceleración de procesamiento de datos transaccionales

Campo técnico

La invención se refiere a sistemas informáticos.

Antecedentes

El crecimiento explosivo en el volumen de datos global, la omnipresencia de los dispositivos que se conectan a las redes, la reducción de las barreras de entrada para creación y compartición de contenido de usuario, la digitalización de muchas funciones anteriormente fuera de línea (banca, recetas médicas, etc.), la aparición de sistemas en red virtualizados y fuera del sitio (nubes), entre otros factores, han contribuido a la aparición de la era de los “grandes volúmenes de datos” (Big Data). Esto presenta desafíos para los sistemas, como procesamiento de aplicaciones que se enfrentan a rendimientos y requisitos de volumen masivo extremo para entregar o distribuir datos procesados a cualquier número de puntos de destino. Estos sistemas pueden ser además escalables para mantener el ritmo con el crecimiento continuo de los grandes volúmenes de datos y posibilitan la interactividad para internet de gran audiencia omnipresente y aplicaciones en la nube.

La utilización de procesadores multi-núcleo ha aumentado drásticamente en la industria informática. En general, el término “procesador” se refiere a la unidad de hardware que lee y ejecuta instrucciones de programa. Históricamente, los procesadores utilizaban originalmente un único “núcleo”, que se refiere a la porción del procesador que lee y ejecuta una secuencia de instrucciones. Un procesador multi-núcleo se refiere a una única unidad de hardware en la que dos o más “núcleos” de procesamiento independientes están integrados en un único paquete. Recientemente, se han hecho disponibles sistemas informáticos que tienen más de 128 a 256 núcleos de procesamiento. Tales plataformas informáticas multi-núcleo presentan desafíos sobre las técnicas de programación tradicional.

El documento US2009/013325 A1 se refiere a un método de asignación de recursos, un programa de asignación de recursos y un aparato de asignación de recursos.

El documento US2009/064156 A1 se refiere a un producto de programa informático y un método para dimensionar la capacidad de entornos virtualizados.

El documento US2009/327669 A1 se refiere a un aparato de procesamiento de información, método de intercambio de programa y medio de almacenamiento.

Sumario

Se definen aspectos de la invención en las reivindicaciones adjuntas.

En general, esta divulgación se refiere a una máquina virtual de software que proporciona aceleración de datos transaccionales de alto rendimiento optimizada para plataformas informáticas multi-núcleo. La máquina virtual utiliza un motor de paralelado subyacente que busca maximizar las eficacias de las plataformas informáticas multi-núcleo para proporcionar una máquina virtual de alto rendimiento (latencia más baja) altamente escalable.

En algunas realizaciones, el motor de paralelización subyacente de la máquina virtual de software proporciona auto-organización en su capacidad para paralelizar y almacenar datos relevantes para procesamiento de transacción a particiones de datos, cada una asociada con diferentes unidades de ejecución para la máquina virtual de software. Además, las tareas que procesan de manera colectiva transacciones y datos transaccionales correspondientes proporcionan auto-ajuste en su capacidad para determinar de manera autónoma y migrar entre unidades de ejecución que procesan las tareas. Como resultado, la máquina virtual de software puede emplear múltiples unidades de delegación de transacción distribuidas y así evitar tanto un administrador de transacción centralizado para gestionar organización de datos como delegación transaccional y los cuellos de botella intrínsecos asociados con tal administración centralizada.

Estas técnicas pueden ser útiles en sistemas requeridos para tratar las necesidades particulares de aceleración de datos dinámica e interactiva para aplicaciones web de gran audiencia y nubes de “grandes volúmenes de datos”. En particular, un sistema que implementa las técnicas descritas puede agregar datos transaccionales para gestionar de manera eficaz la entrada de datos masivos que surgen de muchas fuentes y se reciben mediante el sistema, así como desagregar datos transaccionales para entregar datos procesados para seleccionar destinos. Por ejemplo, una característica de comunicaciones de datos única es la capacidad de la plataforma para enviar interactivamente datos seleccionados de difusión de envío a dispositivos individuales (usuarios) y crear sesiones de difusión privadas interactivas (canales) en un flujo de datos de difusión en masa homogéneo. Las técnicas de la plataforma pueden posibilitar también la aceleración de la transacción y aplicación de mensajería proporcionadas por el cliente en un sistema que opera una caché de base de datos en memoria escalable, con extensión de tiempo de ejecución indefinida en tiempo real bajo demanda integrada para almacenamiento secundario. Un sistema de este tipo puede aprovechar las técnicas de la plataforma para escalar la caché más allá de límites en memoria física y, cuando se

requiera, integrar como parte de los límites de memoria de máquina virtual una extensión automática y uso de dispositivos de memoria físicamente externa (por ejemplo, discos duros). La máquina virtual de software descrita en este punto, en otras palabras, soporta un movimiento desde 'arquitecturas de información estáticas' que tienen dificultades para soportar o crear valor desde grandes volúmenes de datos a un modelo de arquitectura dinámica. Con baja latencia, el procesamiento escalable junto con complejidad reducida y eficacia de costes aumentada, las técnicas descritas tratan específicamente las condiciones del procesamiento de grandes volúmenes de datos para proporcionar la capacidad para consumir y procesar concurrentemente volúmenes de transacciones masivas a partir de grandes números de productores de datos junto con la capacidad para enviar datos procesados a billones de consumidores de datos de una manera interactiva.

En un ejemplo, un dispositivo comprende un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución y una pluralidad de máquinas virtuales que opera cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de los núcleos de ejecución. El dispositivo comprende también una base de datos en memoria que comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución. El dispositivo comprende también una pluralidad de tareas que se ejecutan en las máquinas virtuales para descomponer una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un método comprende ejecutar una pluralidad de máquinas virtuales que operan cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de una pluralidad de núcleos de ejecución de un procesador hardware multi-núcleo de un dispositivo informático. El método comprende también asociar uno diferente de los núcleos de ejecución con cada una de una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria almacena datos para una base de datos en memoria. El método comprende además ejecutar una pluralidad de tareas con las máquinas virtuales, en el que la pluralidad de tareas descomponen una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un dispositivo de almacenamiento legible por ordenador comprende instrucciones que, cuando se ejecutan, provocan que un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución ejecute una pluralidad de máquinas virtuales que opera cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de una pluralidad de núcleos de ejecución de un procesador hardware multi-núcleo de un dispositivo informático. Las instrucciones, cuando se ejecutan, provocan también que el procesador hardware multi-núcleo asocie uno diferente de los núcleos de ejecución con cada una de una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria almacena datos para una base de datos en memoria. Las instrucciones, cuando se ejecutan, provocan además que el procesador hardware multi-núcleo ejecute una pluralidad de tareas con las máquinas virtuales, en el que la pluralidad de tareas descomponen una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

Los detalles de una o más realizaciones de la invención se exponen en los dibujos adjuntos y en la descripción a continuación. Otras características, objetos y ventajas de la invención serán evidentes a partir de la descripción y dibujos y a partir de las reivindicaciones.

Breve descripción de los dibujos

La Figura 1 es un diagrama de bloques que ilustra un sistema de ejemplo que utiliza las técnicas descritas en el presente documento para proporcionar aceleración de datos transaccionales de alto rendimiento.
La Figura 2 es un diagrama de bloques que ilustra una realización de ejemplo de un acelerador de datos transaccionales que proporciona aceleración de datos transaccionales de alto rendimiento de acuerdo con las técnicas descritas en esta divulgación.
La Figura 3 es un diagrama de bloques que ilustra un caso de ejemplo de un motor de paralelización que descompone transacciones o eventos entrantes para procesamiento mediante tareas autónomas que se ejecuta cada una independientemente en un núcleo separado de una plataforma informática multi-núcleo de la manera descrita en esta divulgación.
La Figura 4 es un diagrama de bloques que ilustra tareas que se ejecutan en procesadores virtuales para modificar concurrentemente estructuras de datos almacenadas en caché que usan las técnicas de esta divulgación.
La Figura 5 es un diagrama de bloques que ilustra procesadores virtuales de ejemplo en más detalle.

La Figura 6 es un diagrama de bloques de un sistema que realiza multiplexación por división en el tiempo de tareas de acuerdo con las técnicas descritas en el presente documento.

Las Figuras 7A-7B son diagramas de bloques que ilustran tareas de ejemplo que soportan la operación paralelizada de los aceleradores de datos transaccionales de la Figura 1.

La Figura 8 es un diagrama de bloques que ilustra un acelerador de datos transaccionales de ejemplo que escribe concurrentemente a múltiples objetos de datos en paralelo de acuerdo con las técnicas descritas en el presente documento.

La Figura 9 es un diagrama de bloques que ilustra un acelerador de datos transaccionales 190 de ejemplo que incluye múltiples máquinas de agrupación para escribir estructuras de datos parciales de acuerdo con las técnicas descritas en el presente documento.

La Figura 10 ilustra un sistema de ejemplo en el que se usan instancias del acelerador de datos transaccionales que están acorde con técnicas descritas en el presente documento como bloques de creación para formar una estructura en niveles para enviar interactivamente datos seleccionados a un gran número de clientes individuales (usuarios) y crear sesiones de difusión privadas interactivas (canales) en un flujo de datos de difusión en masa homogéneo.

La Figura 11 ilustra un sistema de ejemplo en el que se usan instancias de un acelerador de datos transaccionales que están acorde con técnicas descritas en el presente documento como bloques de creación para formar una estructura en niveles para recibir datos desde un gran número de fuentes de datos individuales (usuarios).

La Figura 12 es un diagrama de bloques que ilustra un sistema de ejemplo que utiliza las técnicas descritas en el presente documento para proporcionar aceleración de datos transaccionales de alto rendimiento para usuarios que tanto originan como consumen datos.

Descripción detallada

La Figura 1 es un diagrama de bloques que ilustra un sistema de ejemplo que utiliza las técnicas descritas en el presente documento para proporcionar aceleración de datos transaccionales de alto rendimiento. En el sistema de ejemplo 10 de la Figura 1, un conjunto de aceleradores de datos transaccionales 13A-13N (colectivamente, “aceleradores de datos transaccionales 13”) se organizan para formar una agrupación 11 de aceleradores de datos. Los aceleradores de datos transaccionales operan para introducir y procesar continuamente grandes cantidades de transacciones de datos desde fuentes de datos 12 y entregar los datos procesados a las fuentes de datos 16. Las fuentes de datos 12 y el consumidor de datos 16 pueden ser cualquiera que origine o consuma datos, tal como sistemas, aplicaciones o bases de datos.

Como se describe en el presente documento, cada uno de los aceleradores de datos transaccionales 13 incluye un motor de paralelización que proporciona un entorno de operación multi-núcleo masivamente paralelo para que una máquina virtual introduzca y procese transaccionalmente los datos introducidos. La máquina virtual de software proporciona aceleración de datos transaccionales de alto rendimiento optimizados para las plataformas informáticas multi-núcleo subyacentes de los aceleradores de datos transaccionales 13. Es decir, el motor de paralelización proporciona una plataforma de base en la que la introducción y procesamiento de datos pueden paralelizarse de una manera que es altamente escalable y optimizada para ejecución independiente a través de un número arbitrario de núcleos de ejecución de múltiples dispositivos informáticos agrupados. Las transacciones de datos entrantes recibidas desde fuentes de datos 12 se descomponen mediante los motores de paralelización en operaciones que pueden ejecutarse independientemente en los núcleos individuales de las plataformas informáticas multi-núcleo subyacentes de los aceleradores de datos transaccionales 13. Las máquinas virtuales operan en una base de datos en memoria organizada de manera que permite al motor de paralelización de cada uno de los aceleradores ejecutar operaciones transaccionales para insertar, borrar, actualizar y consultar datos desde la base de datos en memoria en paralelo a través de los núcleos del entorno de procesamiento multi-núcleo subyacente.

Además, durante operación, el motor de paralelización de la máquina virtual de software que se ejecuta en cada uno de los aceleradores de datos transaccionales 13 puede auto-organizar dinámicamente el uso de memoria para escalar la base de datos en memoria más allá de los límites en memoria físicos. De esta manera, las máquinas virtuales de software de los aceleradores de datos transaccionales 13 pueden proporcionar una extensión automática y uso de dispositivos de memoria físicamente externos (por ejemplo, discos duros) cuando se procesan los datos de transacción introducidos. Esto permite a las máquinas virtuales aprovechar dinámicamente el almacenamiento virtual para la base de datos en memoria según sea necesario con el procesamiento continuo de las transacciones de entrada en paralelo en el entorno informático multi-núcleo.

Además, las máquinas virtuales ejecutadas mediante los aceleradores de datos transaccionales 13 incluyen características de comunicación paralelizadas que permiten a los aceleradores de datos transaccionales enviar interactivamente datos seleccionados a los consumidores de datos 16. Por ejemplo, los consumidores de datos 16 pueden ser dispositivos individuales, y las características de comunicación paralelizadas de los aceleradores de datos transaccionales 13 pueden crear canales interactivos en un flujo de datos de difusión en masa para enviar grandes cantidades de datos solicitados individualmente a altos volúmenes en los consumidores de datos 16.

De esta manera, los aceleradores de datos transaccionales 13 ejecutan máquinas virtuales que tienen motores de paralelización subyacentes que buscan maximizar las eficacias de las plataformas informáticas multi-núcleo para proporcionar aceleración de transacción de datos de alto rendimiento (latencia más baja) altamente escalable. Además, la máquina virtual puede observarse como una máquina virtual en memoria con una capacidad para auto-organizar en su estado operacional y auto-buscar, en tiempo real, límites de funcionamiento de memoria disponibles para optimizar automáticamente el rendimiento máximo disponible para aceleración de procesamiento de datos y entrega de contenido de cantidades masivas de datos.

Las máquinas virtuales paralelizadas descritas en el presente documento permiten que los datos transaccionales desde las fuentes de datos 12 se determinen dinámicamente al vuelo cuando se dirigen a los consumidores de datos 16 para operaciones de procesamiento adicionales o presentación a usuarios de la forma más apropiada y usable. Además, las máquinas virtuales paralelizadas de los aceleradores de datos transaccionales 13 pueden operar en el flujo de datos en tiempo real sin requerir necesariamente que se escriban los datos en disco y se determinen en su totalidad. Como tal, las máquinas virtuales paralelizadas pueden acelerar la velocidad de procesamiento y la relevancia de los datos transaccionales presentados a los consumidores de datos 16.

Las máquinas virtuales de procesamiento paralelo de los aceleradores de software 13 pueden usarse para aumentar las infraestructuras y aplicaciones de procesamiento de datos existentes en aplicaciones en la nube, móviles y sociales y entornos informáticos empresariales para entregar aceleración de datos altamente escalable de baja latencia con aumento de rendimiento y reducción de costes de operación.

La Figura 2 es un diagrama de bloques que ilustra una realización de ejemplo de un acelerador 13. En el ejemplo ilustrado, el acelerador 13 incluye una máquina virtual 20 que está diseñada específicamente para acelerar y proporcionar procesamiento personalizable para grandes cantidades de datos desde una a muchas fuentes de datos 12. El acelerador 13 se implementa típicamente en una plataforma informática, tal como un servidor de gama alta, que tiene una plataforma informática multi-núcleo 24. La plataforma informática 24 puede incluir cualquier número de procesadores y cualquier número de núcleos hardware desde, por ejemplo, cuatro a miles.

En el ejemplo de la Figura 2, el acelerador 13 incluye una capa de abstracción de plataforma 25 que presenta cierta funcionalidad del núcleo del sistema operativo subyacente 27 para el procesador virtual 20, tal como gestión de memoria y E/S de hardware. En un ejemplo, el procesador virtual 20 puede ejecutarse en un espacio de proceso global proporcionado mediante el núcleo del sistema operativo 27. El núcleo del sistema operativo 27 puede ser, por ejemplo, un Linux, Distribución de Software Berkeley (BSD), otro núcleo variante de Unix o un núcleo de sistema operativo Windows, disponible de Microsoft Corp.

El motor de introducción de datos 26 del procesador virtual 20 opera para “introducir” datos transaccionales entrantes. El motor de introducción de datos 26 puede recibir datos de miles a millones de conexiones de entrada concurrentes, cada dato de flujo continuo hacia dentro sin necesidad de solicitársele nueva información. Los datos pueden introducirse desde unas pocas tuberías “gruesas” o a través de miles de conexiones desde dispositivos o sensores individuales. Los tipos de datos a procesarse pueden ser estructurados, no estructurados o ambos. Por ejemplo, el motor de introducción de datos 26 puede incluir uno o más adaptadores de datos para recibir y procesar datos con formato, por ejemplo, XML y datos con formato CSV.

A medida que los datos entrantes se “introducen” en el sistema mediante el motor de introducción de datos 26, el motor de aceleración de base de datos 28 opera en la base de datos en memoria 27. El motor de aceleración de base de datos 28 proporciona un motor de procesamiento de datos altamente escalable que tiene responsabilidad principal de coordinación de actividades entre el motor de introducción de datos 26, las aplicaciones de cliente 28 y el motor de entrega de contenido 30. En algunos ejemplos, el motor de aceleración de base de datos 28 expone una API basada en SQL convencional mediante la que el motor de introducción de datos 26, las aplicaciones de cliente 28 y el motor de entrega de contenido 30 interactúan con la base de datos en memoria 27 y se ejecutan bajo el control del motor de paralelización 22 de manera que tiene lugar procesamiento a extremadamente baja latencia. En otras palabras, el motor de aceleración de base de datos 28 puede proporcionar una vista al vuelo accesible por SQL en datos entrantes según se introducen y almacenan en la base de datos en memoria 27. En general, el motor de aceleración de base de datos 28 utiliza el motor de paralelización 22 para descomponer transacciones o eventos entrantes en bloques detallados de operaciones que se despliegan a continuación al núcleo de ejecución de hardware más cercano y más disponible relevante para los datos requeridos para procesamiento. El motor de aceleración de base de datos 28 posibilita la descomposición, procesamiento, comprobaciones de concurrencia y re-ensamblaje de transacciones y eventos en resultados calculados.

El motor de entrega de contenido 30 puede emitir datos a uno, varios o muchos destinos, es decir, los consumidores de datos 16. Es decir, el motor de entrega de contenido 30 proporciona la capacidad de que los datos procesados se ‘envíen’ (entreguen) desde la base de datos en memoria 20 como un flujo de datos dirigido a consumidores de datos 16, que pueden ser otros sistemas, aplicaciones o bases de datos. Por ejemplo, en algunas realizaciones, el motor de entrega de contenido puede configurarse para entregar datos mediante un flujo dirigido único a otro sistema informático o almacén de datos. Además, el motor de entrega de contenido 30 puede proporcionar una entrega de

datos bidireccional interactiva y un motor de comunicaciones para difundir interactiva y bi-direccionalmente datos a grandes audiencias o dispositivos, es decir, los consumidores de datos 16.

En algunas realizaciones, el motor de introducción de datos 26 y el motor de entrega de contenido 30 pueden soportar capacidades "seleccionar-convertir" descritas en el presente documento que posibilitan a una fuente de datos 12 o a un consumidor de datos 16 (o ambos) adaptar el contenido que se envía o recibe. Esto puede tomar la forma de canales de datos privados, es decir, flujos de datos interactivos y personalizados únicos para cada fuente o consumidor. Cada conexión puede ser una conexión abierta continua, de manera que los consumidores de datos 16 no necesitan interrogar datos cambiados. El motor de introducción de datos 26 y el motor de entrega de contenido pueden soportar millones de conexiones continuas abiertas a fuentes de datos 12 y consumidores de datos 16.

El motor de paralelización 22 del procesador virtual 20 proporciona un entorno de ejecución abstracto que utiliza arquitecturas de procesadores y placas madre multi-núcleo para formar una plataforma paralela, escalable de baja latencia, altamente integrada para la ejecución del motor de introducción de datos 26, del motor de aceleración de base de datos 28, del gestor de almacenamiento virtual 29 y del motor de entrega de contenido 30. Es decir, el motor de paralelización 22 proporciona una plataforma de ejecución optimizada para sistemas multi-núcleo y de muchos núcleos para proporcionar concurrencia en tiempo real, gestión de memoria y capacidades de gestión de tareas con paralelismo detallado en una base por núcleo. Además, el motor de paralelización 22 asegura uso óptimo de las cachés de instrucciones y de datos (por ejemplo, cachés L1, L2 y L3) mientras que implementa paralelismo detallado descomponiendo todo el procesamiento en tareas que pueden ejecutarse independientemente en núcleos individuales y minimizando el requisito de estructuras de bloqueo concurrente. Esto permite al motor de introducción de datos 26, al motor de aceleración de base de datos 28, al gestor de almacenamiento virtual 29 y al motor de entrega de contenido 30 operar en la máquina virtual 20 con un alto grado de ejecución paralela en la plataforma informática multi-núcleo 24. De esta manera, el motor de paralelización 22 puede observarse como un sistema operativo multi-núcleo masivamente paralelo que proporciona un procesador virtual (máquina virtual 20) para procesar los datos introducidos.

El gestor de almacenamiento virtual 29 de la máquina virtual de software 20 proporciona auto-organización y permite a la máquina virtual escalar más allá de sus límites en memoria física y, cuando se requiera, integrar el uso de dispositivos de memoria físicamente externos (por ejemplo, discos duros). Esto permite a la base de datos en memoria 27 poner en cola a la memoria externa mientras realiza las transacciones para expandir su espacio de memoria para utilizar el almacenamiento persistente 23. Por ejemplo, el gestor de almacenamiento virtual 29 puede poner en cola temporalmente datos transaccionales si los datos no se ajustan en los límites de memoria física y enviar datos fuera para almacenamiento y cálculo. Además, todos estos servicios para gestión de almacenamiento virtual están paralelizados en la máquina virtual 20 y se ejecutan mediante el motor de paralelización para ejecución en la plataforma informática multi-núcleo 24.

Además, el gestor de almacenamiento virtual 29 gestiona el almacenamiento persistente 23 para permitir la recuperación de un fallo o para que los usuarios cierren el sistema y lo recuperen sin pérdidas de datos asociadas. La implementación de persistencia en disco garantiza que no hay pérdidas de transacción en el caso de un fallo. Como se explica en mayor detalle a continuación, las copias de la base de datos en memoria 27 pueden escribirse a ficheros de punto de comprobación en un intervalo de tiempo configurable. Además, en algunos casos, las transacciones pueden grabarse en ficheros de registro diario y la asignación de la transacción únicamente tiene lugar después de que se escriben las entradas en los ficheros de registro diario. Para recuperarse de un fallo de sistema, el subsistema de persistencia del gestor de almacenamiento virtual 29 puede aplicar el último fichero de punto de comprobación y a continuación aplicar todas las entradas de registro diario desde el último punto de comprobación para recrear la base de datos en memoria 27. De esta manera, la persistencia puede implementarse para que sea compatible con ACID (atomicidad, coherencia, aislamiento, durabilidad).

La Figura 3 es un diagrama de bloques que ilustra un caso de ejemplo del motor de paralelización 22 de la Figura 2, que descompone transacciones entrantes o eventos para procesar mediante tareas autónomas cada una ejecutándose independientemente en un núcleo separado de un caso de ejemplo de la plataforma informática multi-núcleo 24 de la manera descrita en el presente documento. En este ejemplo, la plataforma informática multi-núcleo 24 incluye los núcleos de procesamiento 52A-52N ("núcleos 52") que incluyen cada uno una unidad de ejecución independiente para realizar instrucciones que se ajustan a una arquitectura de conjunto de instrucciones para el núcleo. Los núcleos 52 representan un número natural arbitrario de núcleos y cada núcleo está asociado con un índice. Por ejemplo, el núcleo 52A puede asociarse con el índice 1, el núcleo 52B con el índice 2 y así sucesivamente. En un ejemplo, los núcleos 52 representan 256 núcleos (es decir, $N = 256$). Los núcleos 52 pueden implementarse cada uno como circuitos integrados separados (IC) o pueden combinarse en uno o más procesadores multi-núcleo (o procesadores de "muchos núcleos") que se implementan cada uno usando un único IC (es decir, un chip multiprocesador).

Los subconjuntos de los núcleos 52 combinados en un procesador multi-núcleo pueden compartir componentes de procesador mientras que cada núcleo del subconjunto mantiene al menos una unidad de ejecución independiente para realizar instrucciones sustancialmente de manera independiente de los otros núcleos del subconjunto. Por ejemplo, los núcleos 52A, 52B pueden compartir una caché de nivel 3 (L3) y una unidad de gestión de memoria

(MMU) para un procesador multi-núcleo que incluye los núcleos. Sin embargo, los núcleos 52A, 52B en este ejemplo incluyen cada uno una unidad de ejecución separada y cachés de nivel 1 (L1)/nivel 2 (L2) separadas. Como alternativa, los núcleos 52A, 52B pueden compartir cachés L2/L3 y una MMU del procesador multi-núcleo. En algunos casos, la plataforma informática multi-núcleo 24 puede representar una agrupación de placas madre separadas o de tipo cuchilla (blade) de procesamiento (en lo sucesivo, "máquinas de agrupación") insertadas en uno o más chasis. Cada máquina de agrupación en tales casos puede incluir uno o más procesadores multi-núcleo teniendo cada uno un subconjunto de núcleos 52.

Los datos de la base de datos relacional en memoria 27 se almacenan en uno o más medios de almacenamiento legibles por ordenador que incluye las particiones 51A-51N ("particiones 51") localizada cada una en una localización física separada y asociada cada una con uno respectivo de los núcleos 52. El medio de almacenamiento legible por ordenador que almacena la base de datos relacional en memoria puede presentar una arquitectura de acceso de memoria no uniforme (NUMA). Es decir, los núcleos 52 pueden no tener igual tiempo de acceso a memoria para cada una de las particiones 51. En algunos casos, cada una de las particiones 51 asociadas con respectivos núcleos 52 representan la partición de la base de datos relacional en memoria 27 que tiene un tiempo de acceso a memoria que es menor o igual que el tiempo de acceso a memoria a cualquier otra de las particiones del núcleo. En otras palabras, los núcleos 52 pueden usar respectivas particiones 51 que ofrecen latencia de memoria más baja para que los núcleos reduzcan la latencia de memoria global.

Cada una de las particiones 51 comprende medios de almacenamiento legibles por ordenador, tal como medios legibles por ordenador no transitorios que incluyen una memoria tal como memoria de acceso aleatorio (RAM) (incluyendo diversas formas de RAM dinámica (DRAM), por ejemplo, SDRAM DDR2, o RAM estática (SRAM)), memoria Flash, memoria de contenido direccionable (CAM), CAM ternaria (TCAM), u otra forma de medio de almacenamiento fijo o extraíble que puede usarse para llevar o almacenar instrucciones deseadas y datos de la base de datos relacional en memoria 27 y que puede accederse mediante los núcleos 52.

En algunos casos, las particiones 51 pueden representar cada una particiones de un espacio de direcciones físico para un medio de almacenamiento legible por ordenador que se comparte entre uno o más núcleos 52 (es decir, una memoria compartida). Por ejemplo, los núcleos 52A pueden conectarse mediante un bus de memoria (no mostrado) a uno o más paquetes DRAM, módulos y/o chips (tampoco mostrados) que presentan un espacio de direcciones físico accesible mediante el procesador multi-núcleo y almacenar datos para la partición 51A. Mientras la partición 51A puede ofrecer el tiempo de acceso a memoria más bajo al núcleo 52A que cualquiera de las particiones 51, una o más de las otras particiones 51 puede ser directamente accesible al núcleo 52A. En algunos casos, las particiones 51 pueden también, o como alternativa, representar cada una una caché de los núcleos correspondientes 52. Por ejemplo, la partición 51A puede comprender una caché en chip (por ejemplo, una caché L1/L2/L3 o una combinación de las mismas) para el núcleo 52A.

Las particiones 51 almacenan porciones no solapantes de objetos de bases de datos descentralizadas para la base de datos relacional en memoria 27. Tales objetos pueden incluir tablas relacionales o índices almacenados y gestionados usando estructuras de datos subyacentes tales como, por ejemplo, árboles (o "ensayos") de objetos de datos, ficheros planos, zona de variables dinámicas, contenedores de troceo y árboles B+. Como se describe en detalle a continuación, el motor de paralelización 22 distribuye una estructura de datos subyacente separada para respectivos objetos de base de datos a cada una de las particiones 51 y asigna también datos para que se gestionen mediante las estructuras de datos subyacentes para objetos de base de datos a una diferente de las particiones 51, paralelizando de manera eficaz los objetos de la base de datos entre las particiones. Puesto que cada uno de los núcleos 52 lee y escribe desde una diferente de las particiones 51, las particiones 51 no se someten a corrupción debido a la operación concurrente de múltiples núcleos 52. Como resultado, las tareas que se ejecutan en los núcleos 52 pueden evitar bloquear las particiones 51 en muchas circunstancias.

La interfaz de cliente 41 del motor de paralelización 22 presenta una interfaz mediante la que los clientes pueden emitir solicitudes al acelerador 13. En algunos aspectos, la interfaz de cliente 41 implementa los conectores (sockets) de capa de transporte (por ejemplo, Protocolo de Control de Transmisión (TCP)/Protocolo de Internet (IP) o Protocolo de Datagramas de Usuario (UDP)/IP) para recibir y devolver datos desde/a clientes que invocan la interfaz.

El motor de paralelización 22 descompone la transacción entrante 40 en sub-transacciones detalladas 42A-42N ("sub-transacciones 42") y distribuye las sub-transacciones a múltiples tareas de ejecución que se ejecutan en el uno de los núcleos 52 que está asociado lógicamente con la una de las particiones 51 relevante a los datos para las respectivas sub-transacciones 42. En algunos casos, la una relevante de las particiones 51 es la partición que almacena datos para devolverse en una transacción de tipo consulta 40. En algunos casos, la una relevante de las particiones 51 es la partición que almacena, para un objeto de índice de base de datos, una estructura de datos subyacente que es para almacenar los datos para una transacción de tipo insertar 40 que hace referencia al objeto de índice de base de datos.

El motor de paralelización 22 puede determinar automáticamente un número de núcleos 52 del acelerador 13 sin requerir configuración de software por un administrador. Tras una determinación del número de núcleos 52, el motor

de paralelización crea una correspondiente de las particiones 51 para cada uno de los núcleos 52. Esta característica puede permitir despliegue paralelo para un número arbitrario de núcleos 52, de nuevo, sin requerir reconfiguración del software subyacente.

La transacción 40 es una unidad de transacción, es decir, una unidad de funcionamiento auto-contenida recibida y realizada mediante el acelerador 13 para alterar un estado de la base de datos relacional en memoria 27. La transacción 40 puede ser compatible con ACID para proporcionar aislamiento entre las transacciones para ejecución concurrente y para proporcionar retroceso en el caso de fallo. La transacción 40 puede incluir una cadena de solicitud que se ajusta, por ejemplo, a una sentencia de lenguaje declarativo, un lenguaje de consulta o programa de lenguaje de programación de consultas, un programa de lenguaje de programación funcional o programa de lenguaje de reglas que especifica la respectiva unidad de funcionamiento para realizarse mediante el acelerador 13. La transacción 40 contiene una o más subunidades de funcionamiento que pueden realizarse mediante una unidad de ejecución independiente como sub-transacciones individuales 42 de la transacción padre atómica 40. Las sub-transacciones 42 pueden incluir, con respecto a la base de datos relacional en memoria 27; leer, escribir, manipular y borrar datos; crear y gestionar objetos de base de datos; crear y gestionar metadatos; y operaciones aritméticas y de manipulación de cadenas.

La tarea de compilador 44 (“compilador 44”) recibe la transacción 40 y descompone las transacciones a sub-transacciones 42 usando la gramática de transacción 47 (“gramática 47”), que describe el lenguaje particular de transacciones entrantes, incluyendo la transacción 40, en combinación con el esquema de base de datos 45 para la base de datos relacional en memoria 27 y la biblioteca de etapas 49. En un ejemplo, la gramática 47 incluye un conjunto de una o más reglas de sustitución teniendo cada una variables para coincidir datos en el esquema de base de datos 45 que describen la organización de la base de datos relacional en memoria 27. El esquema de la base de datos 45 puede comprender un diccionario de datos. Cada una de las reglas de sustitución de la gramática 47 hace referencia a variables para reglas de sustitución adicionales en la gramática 47 o etapas en la biblioteca de etapas 49. El compilador 44 analiza la transacción 40 para generar cadenas de solicitud en testigo, y a continuación el compilador 44 aplica la gramática 47 a cada cadena de solicitud en testigo en vista del esquema de la base de datos 45 para producir, para la transacción, una o más series de etapas almacenadas mediante la biblioteca de etapas 49. Cada serie de etapas constituye una tarea separada que, cuando se ejecutan en serie mediante una tarea de ejecución, realizan una de las sub-transacciones 42. De esta manera, el compilador 44 descompone la transacción 40 a sub-transacciones 42 para distribución a y ejecución mediante múltiples núcleos 52.

La biblioteca de etapas 49 comprende un diccionario que mapea grupos de instrucciones de etapas ejecutables mediante los núcleos 52 a claves de etapa referenciadas mediante la gramática 47. Cada grupo de instrucciones de etapa puede incluir instrucciones ejecutables por máquina pre-compiladas para los núcleos 52. Para ejecutar una serie de etapas (es decir, una tarea) cada una identificada mediante una clave de etapa, una unidad de ejecución mapea las claves de etapa para las etapas a la biblioteca de etapas 49 para obtener las instrucciones de etapa mapeadas correspondientes, a continuación ejecuta las instrucciones de etapa mapeadas en una base de etapa a etapa. Cada una de las tareas 44, 46, y las tareas de ejecución subordinadas 48A-48N (ilustradas como “sub-ejecución” 48A-48N) representan series respectivas de etapas para las sub-transacciones 42 correspondientes.

Habiendo descompuesto la transacción 40 en sub-transacciones 42, el compilador 44 provoca la tarea de ejecución 46 para gestionar la ejecución de las sub-transacciones y devolver cualquier respuesta requerida para la transacción 40. De esta manera, el compilador 44 genera un plan de ejecución y provoca la tarea de ejecución 46 para realizar el plan de ejecución. La tarea de ejecución 46 provoca las tareas de ejecución subordinadas 48 para ejecutar sub-transacciones 42 correspondientes. En algunos casos, la transacción 40 puede representar múltiples transacciones de base de datos separadas. En tales casos, el compilador 44 puede producir una tarea de ejecución separada para gestionar cada transacción o reusar la tarea de ejecución 46 para gestionar las transacciones.

Las sub-transacciones 42 pueden cada una hacer referencia a diferentes datos almacenados mediante la base de datos relacional en memoria 27 en particiones separadas 51. Por ejemplo, la transacción 40 puede comprender una solicitud de consulta para filas de una tabla de base de datos que tienen valores de campo que coinciden con múltiples criterios, por ejemplo, una sentencia de SQL SELECT con un apartado WHERE, donde la tabla de la base de datos tiene un índice correspondiente definido para el campo. Como otro ejemplo, la transacción 40 puede comprender una solicitud para añadir una fila a una tabla de la base de datos que tiene múltiples índices definidos, por ejemplo, una sentencia de SQL INSERT, o para actualizar con un nuevo valor de campo todas las filas de la tabla de la base de datos que coinciden con uno o más criterios, por ejemplo, una sentencia de SQL UPDATE. Como otro ejemplo, la transacción 40 puede comprender una solicitud para devolver una suma de todos los valores para un campo de fila de una tabla de la base de datos. La tarea de ejecución 46 produce tareas de ejecución subordinadas 48 para sub-transacciones 42 y asigna las tareas a diferentes núcleos 52 basándose en los datos relacionados. La tarea de ejecución 46 puede proporcionar un puntero de memoria a sí misma para posibilitar que las tareas de ejecución subordinadas 48 devuelvan datos resultantes o información de estado. Cualquiera de las tareas de ejecución subordinadas 48 pueden a su vez producir tareas de ejecución subordinadas adicionales en una descomposición recursiva de las sub-transacciones 42.

La tarea de ejecución 46 introduce datos para una sub-transacción 42 a un algoritmo de asignación, tal como una función de troceo, que emite un índice u otro identificador que identifica uno de los núcleos 52. Para la solicitud para añadir un ejemplo de fila, anteriormente, la tarea de ejecución 46 puede introducir al algoritmo de asignación los datos de fila completos o un subconjunto de los datos de fila, tal como los datos de fila para campos para los que se definen índices para la tabla de la base de datos. El algoritmo de asignación puede ser, por ejemplo, un MD5, SHA-1, o una operación a nivel de bits aplicada a los datos de entrada y el módulo del número de los núcleos 52, o cualquier otra función que produzca un valor en un intervalo del número de núcleos cuando se proporcionan datos de entrada arbitrarios. Para los ejemplos de solicitud de consulta y sentencia de actualización, anteriormente, la tarea de ejecución 46 puede trocear los criterios para la solicitud de consulta y a continuación calcular el módulo de la salida del troceo del número de núcleos. El índice de núcleo emitido mediante el algoritmo de asignación, datos de objeto de base de datos proporcionados para las sub-transacciones 42, determina la ejecución de uno de los núcleos 52 de las respectivas tareas de ejecución subordinadas 48 para las sub-transacciones cuando se producen mediante la tarea de ejecución 46. En el ejemplo ilustrado, el núcleo 52A que tiene el índice 1 ejecuta la sub-transacción 42A puesto que los datos relacionados con la sub-transacción 42A provoca que el algoritmo de asignación emita el índice 1. El núcleo 52B que tiene el índice 2 ejecuta las sub-transacciones 42B puesto que los datos relacionados con la sub-transacción 42B provocan que el algoritmo de asignación emita el índice 2, y así sucesivamente. De esta manera, el algoritmo de asignación asocia datos con diferentes particiones 51 y también con los núcleos 52 que acceden a las respectivas particiones cuando delegan las sub-transacciones 48 de acuerdo con el algoritmo de asignación. Para el ejemplo de la suma de todos los valores, anteriormente, la tarea de ejecución 46 produce una de las sub-transacciones 42 para cada núcleo 52. Cada una de las sub-transacciones provoca respectivas tareas de ejecución sub-ordinadas 48 para calcular, una suma parcial de datos para la tabla de la base de datos almacenada mediante la una asociada de las particiones 51.

Cada uno de los núcleos 52 opera en una diferente de las particiones 51. Dirigiendo las tareas de ejecución subordinadas 48 a diferentes núcleos 52 para ejecución, la tarea de ejecución 46 produce datos relacionados con las respectivas sub-transacciones 42 para las tareas de ejecución subordinadas 48 para almacenarse mediante diferentes particiones conocidas 51. En el ejemplo ilustrado, puesto que el núcleo 52B asociado lógicamente con la partición 51B ejecuta la tarea de ejecución subordinada 48B, la partición 51B almacena datos relacionados con la sub-transacción 42B. Las técnicas por lo tanto paralelizan eficazmente las transacciones así como asignan tareas de ejecución a los núcleos 52 que ofrecen un tiempo de acceso a memoria más bajo a las particiones 51 que almacenan los datos paralelizados relacionados con las respectivas tareas de ejecución. Las tareas de ejecución subordinadas 48 pueden migrar entre los núcleos 52 cuando una correspondiente de las sub-transacciones 42 hace referencia a datos almacenados mediante múltiples particiones 51. De esta manera, las tareas se ejecutan más cercanas, en latencia de memoria, a los datos requeridos por las tareas.

Cada una de las tareas de ejecución subordinadas 48 comprende una serie de etapas. Para ejecutar las etapas, las tareas de ejecución subordinadas 48 pueden mapear las claves de etapa para las etapas a instrucciones de etapa correspondientes en la biblioteca de etapas 49 y dirigir los núcleos 52 para ejecutar las instrucciones de etapa. Cada una de las tareas de ejecución subordinadas 48 se ejecuta en uno diferente de los núcleos 52. Las tareas de ejecución subordinadas 48 pueden ejecutarse por lo tanto sustancialmente en paralelo a pesar de realizar, en combinación, una única transacción 40. Como resultado, el motor de paralelización 22 puede conseguir una mejora sustancial en velocidad de procesamiento de transacción que escala a un número arbitrario de núcleos 52 y al nivel de paralelización intrínseco en la transacción 40. Además, el motor de paralelización 22 consigue tal paralelización en un único sistema con una base de datos relacional en memoria 27 consolidada que puede accederse independientemente mediante múltiples núcleos 52 del sistema, en lugar de en un sistema distribuido que divide una base de datos entre múltiples servidores de bases de datos y por lo tanto requiere un servidor de equilibrio de carga separado o un controlador para equilibrar los datos de la base de datos entre las múltiples particiones.

Las tareas de ejecución subordinadas 48 pueden proporcionar respectivos valores de retorno a la tarea de ejecución 46 referenciados mediante un puntero de memoria a la tarea de ejecución 46 referenciada con las tareas de ejecución subordinadas. Los valores de retorno pueden incluir, por ejemplo, datos solicitados, datos parciales (por ejemplo, una suma parcial), y valores de estado de ejecución (por ejemplo, satisfactorio, fallo). La tarea de ejecución 46 genera una cadena de respuesta para la transacción 40 usando los valores de retorno y emite la cadena de respuesta a uno o más clientes solicitantes mediante la interfaz de cliente 41, o la tarea de ejecución 46 agrega los valores de retorno para procesamiento adicional con otra tarea en el motor de paralelización 22.

Aunque se han descrito con respecto a operaciones realizadas en una base de datos en memoria, las técnicas de esta divulgación se aplican a otras aplicaciones que pueden beneficiarse del procesamiento paralelizado de transacciones entrantes. Por ejemplo, la transacción 40 puede representar una unidad de datos de paquetes (PDU) que tiene una pluralidad de datos de campo que deben procesarse individualmente. Estos datos de campo pueden descomponerse mediante el compilador 44 en un número de sub-transacciones para ejecución mediante las respectivas tareas de sub-ejecución 48. En algunos casos, la transacción 40 puede representar un fragmento de código que se ajusta a un lenguaje de programación, tal como C/C++ o Java. En tales casos, el compilador 44 puede ejecutar un compilador para que el lenguaje de programación produzca dinámicamente código máquina para ejecución mediante las tareas de sub-ejecución 48 directamente en los respectivos núcleos 52 para procesar transacciones de entrada (por ejemplo, paquetes). De esta manera, el motor de paralelización 22 puede alterar

dinámicamente el programa de ejecución de acuerdo con los fragmentos de código recibidos para soportar procesamiento paralelizado flexible de datos de entrada/salida (por ejemplo, PDU).

La Figura 4 es un diagrama de bloques que ilustra una vista conceptual en la que un procesador virtual 20 puede observarse como una pluralidad de procesadores virtuales cooperativos 84 que se ejecutan en respectivos núcleos de la plataforma informática multi-núcleo 24. Como se muestra en el ejemplo de la Figura 4, cada uno de los procesadores virtuales 84 ejecuta tareas 82 en paralelo para modificar concurrentemente estructuras de datos almacenadas en caché usando las técnicas de esta divulgación y realiza la diversa funcionalidad externa e interna del acelerador 13. Cada uno de los procesadores virtuales 84 proporciona entornos de operación pseudo-independientes para planificar y gestionar la ejecución de un conjunto correspondiente de tareas 82A-82N a través de respectivos núcleos de una plataforma informática multi-núcleo 24.

Los procesadores virtuales 84 pueden ofrecer un entorno de ejecución independiente de plataforma uniforme para gestión de procesador virtual; planificación de tareas; compilación y ejecución de sentencias; procesamiento de transacción de base de datos; registro diario; equilibrio de carga de procesador virtual; persistencia, recuperación y replicación de base de datos; introducción y salida de datos; y acciones definidas por el usuario. Los procesadores virtuales 84 pueden implementar el entorno de ejecución ofreciendo una arquitectura de conjunto de instrucciones virtual (ISA) que es uniforme para cada uno de los procesadores virtuales. Los procesadores virtuales 84 reciben instrucciones para ejecutar tareas y traducir las instrucciones a llamadas de biblioteca a nivel de núcleo y/o a instrucciones que se ajustan a la ISA nativa proporcionada mediante los respectivos núcleos. De esta manera, los procesadores virtuales proporcionan un conjunto de máquinas virtuales completamente paralelizadas con las que ejecutar las tareas 82.

En algunas realizaciones, una pluralidad de núcleos 52 soporta una única instancia de un núcleo y un proceso para proporcionar un procesador virtual 20. Por ejemplo, los núcleos 52A, 52B pueden ejecutar subprocesos para un único proceso. En tales realizaciones, los núcleos 52 que cooperan para proporcionar un procesador virtual tienen acceso a un espacio de direcciones físico o virtual único proporcionado mediante el procesador. Tales procesadores virtuales pueden denominarse también como máquinas virtuales de proceso. Como resultado, las tareas que se ejecutan en los núcleos cooperativos pueden pasar mensajes, migrar y producir otras tareas entre los núcleos escribiendo en y leyendo desde el espacio de direcciones común.

Las tareas 82 incluyen un conjunto de etapas que se ajustan al entorno de programación independiente de plataforma uniforme proporcionado mediante cada uno de los procesadores virtuales. Las etapas pueden representar una realización de ejemplo de las etapas de la biblioteca de etapas 49 descrita con respecto a la Figura 3. Es decir, los procesadores virtuales 84 pueden traducir etapas a un conjunto de instrucciones ejecutables por máquina mediante los núcleos 52. Como resultado, cada una de las tareas 82 puede migrar sin interrupciones y ejecutarse en cualquiera de los procesadores virtuales 84 sin requerir volver a compilar a una nueva ISA o traducción a un nuevo entorno de programación.

Las cachés 92 de los respectivos núcleos 52 almacenan datos asociados con el respectivo núcleo y pueden representar realizaciones de ejemplo de las particiones 51 de la Figura 3. Cada una de las cachés 92 incluye una de las estructuras parciales 94A-94N (“estructuras parciales 94”) que almacenan en caché datos para una estructura de datos colectiva que representa un objeto de base de datos, tal como el índice 93 definido para la tabla 98 de la base de datos relacional en memoria 27. En otras palabras, cada una de las estructuras parciales 94 almacena en caché un subconjunto no solapante de los datos para el índice 93. Las estructuras parciales 94 y el índice 93 pueden incluir, por ejemplo, tablas, listas vinculadas y árboles B+. De acuerdo con las técnicas de esta divulgación, las estructuras parciales 94 almacenan en caché datos respectivos para el índice 93 cuando los datos, cuando se introducen a un algoritmo de asignación ejecutado mediante uno de los procesadores virtuales 84, dan como resultado un valor de índice asociado con uno de los núcleos 52 que comprende la estructura parcial en su caché. Las estructuras parciales 94 pueden incluir subconjuntos de datos almacenados y gestionados mediante cualquier objeto de base de datos que pueda dividirse, incluyendo tablas, índices, filas de tablas individuales y estructuras internas. Además, aunque se ilustra como que residen en las cachés 92, las estructuras parciales 94 pueden describirse en uno cualquiera o más medios de almacenamiento legible por ordenador.

Por ejemplo, la estructura de datos colectiva puede ser un índice que incluye valores de campo para el campo de índice de base de datos que se mapean cada uno a un puntero que resuelve a una fila de la tabla 98 almacenada en la base de datos relacional en memoria 27. En este ejemplo, cada una de las estructuras parciales 94 incluye mapeos de valor-puntero de campo asignados a uno de los núcleos 52 que incluye la estructura parcial. Como resultado, las tareas 82 que se ejecutan en los procesadores virtuales 84 pueden determinar rápidamente la localización de los mapeos de valor-puntero de campo entre las estructuras parciales 94 para el índice de la base de datos troceando datos de valor de campo. En algunos casos, los núcleos 52 pueden mantener una matriz u otra estructura de datos asociativa para el índice 93 que mapea valores de índice para los núcleos 52 a direcciones de memoria, en espacio de memoria, para las correspondientes estructuras parciales 94. Tras aplicar el algoritmo de asignación para determinar un valor de índice para buscar datos, las tareas 82 mapean el valor de índice a la dirección de memoria para una de las estructuras parciales 94 y migran al procesador virtual 84 que corresponde a uno de los núcleos 52 asociado con el valor de índice para ejecución. De esta manera, cada uno de los núcleos 52 está asociado lógicamente con los datos en las respectivas estructuras parciales 94, y las técnicas pueden mejorar

el rendimiento de caché aumentando la probabilidad de que las estructuras parciales 94 permanezcan en la una correspondiente de las cachés 92. En algunos casos, una tarea especializada gestiona la estructura de datos asociativa para las tareas de ejecución.

En el ejemplo ilustrado, los núcleos 52 comunican para intercambiar datos, mensajes y tareas mediante el bus de sistema 98. Además, los núcleos 52 hacen de interfaz con la memoria de sistema 99, incluyendo la base de datos relacional en memoria 27, mediante el bus de memoria 99. Los procesadores virtuales 84 ejecutan por separado las tareas 82 en paralelo para realizar la diversa funcionalidad externa e interna del acelerador 13. Las tareas 82 pueden ser punteros que resuelven a una estructura de tarea en la memoria de sistema 99 que incluye una serie de etapas para ejecución mediante los procesadores virtuales 84. Las tareas 82 pueden por lo tanto identificarse de manera única mediante su dirección en el espacio de direcciones de la memoria de sistema 99. Cada una de las tareas 82 se ejecuta sustancialmente de manera independiente de cada una de las otras tareas 82. Mientras las tareas 82 pueden intercambiar datos con otras tareas, producir tareas adicionales y producirse desde otras tareas, cada una de las tareas 82 auto-determina el uno de los núcleos 52 que va a ejecutar tarea. No hay tareas o procesos de supervisión para especificar una localización de núcleo para las tareas 82. Esta heterarquía de tareas cooperativas 82 está por lo tanto auto-dirigida y auto-organizada, reduciendo sustancialmente el número de ciclos de los núcleos 52 dedicados a gestión de tareas, comprobación de coherencias y otras funciones administrativas.

Las tareas 82 pueden migrar entre los procesadores virtuales 84 y producir tareas adicionales para ejecutarse en otros procesadores virtuales 84. En el ejemplo ilustrado, la tarea 82A que se ejecuta en el procesador virtual 84A produce la tarea 82B para ejecutarse en el procesador virtual 84N enviando el mensaje 83 al procesador virtual 84N. El mensaje 83 puede especificar una serie de etapas determinadas mediante la tarea 82A para la tarea producida 82B. El mensaje 83 puede como alternativa especificar un puntero que resuelve a una estructura de tarea en la memoria de sistema 99 que incluye una serie de etapas para ejecutar como la tarea 82B. Además, la tarea 82A se copia posteriormente a sí misma para ejecutarse en el procesador virtual 84B enviando el mensaje 85 al procesador virtual 84B. El mensaje 85 puede especificar una serie de etapas que representa una restante de la tarea 82A que requiere ejecución o un puntero que resuelve a una estructura de tarea en la memoria de sistema 99 que incluye al menos una serie restante de etapas para ejecución mediante el procesador virtual 84B.

La Figura 5 es un diagrama de bloques que ilustra los procesadores virtuales 84A-84N, con detalle adicional, que ejecutan múltiples tareas paralelas de acuerdo con las técnicas de esta divulgación. Cada uno de los procesadores virtuales 84 está asociado con una de las cachés 92A-92N puesto que el procesador virtual se ejecuta en un núcleo que incluye la caché asociada. Aunque los componentes y funcionalidad de los procesadores virtuales 84 se describen como alternativa con respecto a unos individuales de los procesadores virtuales, cada uno de los procesadores virtuales 84 incluye sustancialmente componentes similares para realizar sustancialmente funcionalidad similar. En algunos casos, múltiples núcleos pueden ejecutar subprocesos para un proceso que proporciona uno de los procesadores virtuales 84. En tales casos, el proceso incluye conjuntos de componentes separados para cada uno de los núcleos. Por ejemplo, un único procesador virtual 84 en tales casos puede proporcionar cuatro instancias del conjunto de una lista de ejecución 104, lista en espera 108, cola-cruzada 110 y tarea de indicación de funcionamiento 102. En tales casos, el planificador 100 para el proceso se ejecuta en núcleos separados para llevar a cabo las tareas 82 en el proceso. La referencia en el presente documento a un procesador virtual puede por lo tanto como alternativa hacer referencia a un proceso virtual y a uno de los conjuntos de componentes proporcionados en el presente documento.

La lista de ejecución 104A del procesador virtual 84A almacena una lista de tareas puestas en cola actualmente para ejecución mediante el procesador virtual. En el ejemplo ilustrado, la lista de ejecución 104A es una cola circular que almacena punteros de memoria que se resuelven a estructuras de tareas respectivas en el espacio de memoria para el procesador virtual 84A. La lista de ejecución 104A, como otras estructuras de datos que soportan el procesador virtual 84A, pueden almacenarse en la caché 92A y/o en memoria principal. El planificador 100A invoca iterativamente tareas en la lista de ejecución 104A. El planificador 100A realiza la multiplexación por división en el tiempo con divisiones de tiempo variable que dependen de las instrucciones en las tareas. El planificador 100A puede producir subprocesos separados para ejecutar cada uno una tarea en la lista de ejecución 104A. Como alternativa, el planificador 100A puede usar un único subproceso que funciona para la lista de ejecución 104A. Además de los subprocesos para ejecutar la lista de ejecución 104A, el planificador 100A puede usar subprocesos adicionales para realizar tareas especializadas. El planificador 100A invoca una tarea de la lista de ejecución 104A para ejecutar una división de tiempo, a continuación invoca una siguiente tarea de 104A. Puesto que la lista de ejecución 104A es una cola circular, el planificador 100A ejecuta iterativamente las tareas de la lista de ejecución desde la cabecera de la lista a la cola de la lista a continuación, tras finalizar la ejecución, al menos una porción de la tarea en la cola de la lista, ejecuta de nuevo la tarea en la cabecera de la lista.

Las tareas migran entre los procesadores virtuales 84 de manera que una tarea que se ejecuta inicialmente en uno de los procesadores virtuales 84 puede ejecutarse más tarde en otro procesador virtual. Además, una tarea que se ejecuta en uno de los procesadores virtuales 84 puede producir una nueva tarea para ejecución en otro de los procesadores virtuales. En el ejemplo ilustrado, la tarea 112 migra desde el procesador virtual 84A al procesador virtual 84B añadiendo un puntero de memoria para sí misma en la cola cruzada 110B en el mensaje 114 que comprende, en este ejemplo, una operación de escritura de memoria. Las listas de ejecución 104 de los

procesadores virtuales 84 pueden accederse en cualquier momento, y, con la excepción de las operaciones que implican tareas de indicación de funcionamiento 102, los procesadores virtuales 84 se ejecutan independientemente en paralelo y no sincronizan su ejecución de tareas. En algunos casos, los procesadores virtuales 84A, 84B pueden ejecutarse en máquinas de agrupación separadas. Como resultado, ninguno de los procesadores virtuales 84A, 84B pueden acceder al espacio de memoria físico del otro. En tales casos, el mensaje 114 puede incluir un mensaje basado en red tal como, por ejemplo, una escritura de conector, o un mensaje transversal, de panel posterior u otro de conmutación.

Los planificadores 100 pueden migrar las tareas 82 entre los procesadores virtuales 84 debido a una arquitectura NUMA del procesador virtual 20, ejecutando los procesadores virtuales 84 en los núcleos 52 que tienen tiempos de acceso a memoria no uniformes a las cachés 92. De esta manera, los planificadores 100 pueden proporcionar planificación a nivel de NUMA para reducir la latencia global para los accesos a memoria y mejorar de esta manera el rendimiento.

Para evitar la corrupción de las listas de ejecución 104 debido a una adición asíncrona de una nueva tarea, los procesadores virtuales 84 incluyen respectivas colas-cruzadas 110 que almacenan temporalmente cero o más nuevas tareas para añadir a las listas de ejecución 104. En operación, la tarea 112 que se ejecuta en el procesador virtual 84A determina que puede operar más eficazmente en el procesador virtual 84B y se migra a sí misma al procesador virtual 84B bloqueando la cola cruzada 110B y enviando un puntero de memoria para la tarea 112 a la cola cruzada. Para producir una nueva tarea en el procesador virtual 84B, la tarea 112 que se ejecuta en el procesador virtual 84A puede crear una nueva estructura de datos de tarea en memoria y a continuación enviar un puntero de memoria a la nueva estructura de datos de tarea a la cola cruzada 110B. El planificador 100B se ejecuta en el procesador virtual 84B para hacer aparecer la tarea de cabecera de la cola cruzada 110B e insertar la tarea aparecida en lista de ejecución 104B. Utilizando colas cruzadas 110 de esta manera, los procesadores virtuales 84 pueden evitar bloquear las respectivas listas de ejecución 104 para leer/escribir las listas de ejecución evitando además colisiones debido a la ejecución de tareas concurrentes y migración/producción mediante procesadores virtuales separados que se ejecutan en paralelo. En algunos casos, para reducir la posibilidad de colisiones con respecto a colas cruzadas 110A, el procesador virtual 84A puede incluir múltiples colas cruzadas, por ejemplo, una cola cruzada por procesador virtual en el sistema.

En algunos casos, la tarea 112 puede migrar al procesador virtual 84B puesto que un algoritmo de asignación ejecutado mediante la tarea determina que la tarea 112 requiere acceso a un objeto en la estructura parcial 94B de la caché 92B asociada con el procesador virtual 84B. Como se ha descrito anteriormente con respecto a la Figura 4, las estructuras parciales 94 almacenan un subconjunto de datos para un objeto de base de datos global para la base de datos relacional en memoria 27. En algunos casos, las estructuras parciales 94 pueden representar estructuras parciales alternativa o adicionalmente almacenadas en la memoria principal. Para evitar bloquear las estructuras parciales 94 durante acceso mediante las tareas que se ejecutan en los procesadores virtuales 84, el acceso a las respectivas estructuras parciales puede limitarse a tareas que se ejecutan en uno de los procesadores virtuales 84 asociado con la estructura parcial. La tarea 112 debe por lo tanto operar en el procesador virtual 84B para acceder a la estructura parcial 94B. Esta restricción asegura que el acceso a las estructuras parciales 94 por las tareas es seguro y fiable incluso aunque las tareas eviten, en muchas circunstancias, bloquear las estructuras parciales e incluso aunque múltiples diferentes tareas puedan compartir el objeto de base de datos global. Además, múltiples tareas que se ejecutan en diferentes procesadores virtuales 84 pueden acceder al objeto de base de datos global concurrentemente accediendo por separado a diferentes estructuras parciales 94 que juntas constituyen el objeto de base de datos. Cuando, sin embargo, la tarea 112 no puede completar el acceso a uno de los recursos parciales 94 en su división de tiempo permitida, la tarea 112 puede bloquear el recurso parcial para asegurar que los datos del recurso parcial permanecen estables y coherentes para la tarea hasta su siguiente división de tiempo. Como alternativa, la tarea 112 puede bloquear únicamente un elemento almacenado mediante una de las estructuras parciales 94, en lugar de la estructura completa. De esta manera, una tarea posterior puede modificar cualquier elemento no bloqueado de la estructura parcial.

En algunos casos, las tareas de la lista de ejecución 104A pueden requerir recursos no inmediatamente disponibles o estar esperando de otra manera la satisfacción de una dependencia para continuar la ejecución. Para evitar congestionar el núcleo asociado que ejecuta el procesador virtual 84A, tales tareas pueden esperar añadiéndose a sí mismas a la lista de espera 108A junto con un tiempo de activación asociado. La lista de espera 108A almacena tareas en espera ordenadas por tiempo de activación en una estructura de datos ordenada, tal como una cola, tabla, lista vinculada o estructura de datos en árbol. Cada nodo en la lista de espera 108A es por lo tanto un puntero de memoria a una estructura de tarea para una tarea en espera.

Una tarea de alarma y un temporizador de hardware para el procesador virtual 84A gestionan tareas en espera en la lista de espera 108A. La tarea de alarma programa el temporizador de hardware con un valor de tiempo de activación para la tarea más reciente en la lista de espera 108A. Cuando se enciende el temporizador de hardware, se desencadena la tarea de alarma y añade la tarea más reciente en la lista de espera 108A a lista de ejecución 104A. En algunos casos, la tarea de alarma modifica la lista de ejecución 104A para asegurar que el planificador 100A invoca la tarea más reciente siguiente entre las tareas en la lista de ejecución. La tarea de alarma a continuación reprograma el temporizador de hardware con un valor de tiempo de activación para la siguiente tarea

más reciente de acuerdo con la lista de espera 108A. El temporizador de hardware puede accionarse con un reloj de CPU que tiene una velocidad que supera 1 GHz y por lo tanto tiene periodicidad por debajo del microsegundo. Como resultado, la tarea de alarma junto con el temporizador de hardware pueden conseguir la gestión de espera de tareas detallada y el comportamiento de operación del procesador virtual, y pueden por lo tanto mejorar la utilización de recursos asegurando que las tareas se activan y ejecutan en una latencia corta después de su valor de tiempo de activación asociado.

Los procesadores virtuales 84 ejecutan respectivas tareas de indicación de funcionamiento 102A-102N ("tareas de indicación de funcionamiento 102") a una velocidad predefinida para sincronizar una posición operacional de los procesadores virtuales una vez para cada periodo definido mediante la velocidad de indicación de funcionamiento. En algunos casos, la velocidad predefinida es 1 Hz. Por ejemplo, el planificador 100A invoca tareas de la lista de ejecución 104A y, una vez por segundo, ejecuta la tarea de indicación de funcionamiento 102A. Para sincronizar la posición operacional de los procesadores virtuales 84, las tareas de indicación de funcionamiento 102 pueden acceder cada una y disminuir una variable atómica compartida entre todas las instancias de los procesadores virtuales. La variable atómica puede inicializarse con un número de procesadores virtuales 84 (que corresponde al número de núcleos en el sistema). Cada una de las tareas de indicación de funcionamiento 102 prueba la variable atómica para cero. Cuando la variable atómica no es cero, las tareas de indicación de funcionamiento esperan una señal. Cuando la variable atómica alcanza cero debido a la operación de la tarea de indicación de funcionamiento final para el ciclo particular, la tarea de indicación de funcionamiento final puede inicializar una o más tareas de nivel de usuario o señalizar cada uno de los procesadores virtuales 84 para reanudar la ejecución de sus respectivas listas de ejecución 104. De esta manera, la tarea de indicación de funcionamiento final cambia la fase de todas las tareas a la indicación de funcionamiento (es decir, el tiempo de la señal de la señal de todo el sistema) de la tarea de indicación de funcionamiento final. Las tareas de indicación de funcionamiento 102 proporcionan por lo tanto una ventana de tiempo en la que se conoce el estado de cada procesador virtual 84. Las tareas pueden aprovechar esta ventana de tiempo para realizar operaciones de todo el sistema.

Por ejemplo, las tareas pueden establecer un gancho de tarea (por ejemplo, un puntero de memoria a una estructura de tarea en memoria) en cada una de las tareas de indicación de funcionamiento 102. Tras recibir una señal desde la tarea de indicación de funcionamiento final para un ciclo, cada una de las tareas indicación de funcionamiento espera que la señal comience la ejecución y ejecuta la tarea enganchada. La tarea enganchada, cuando se ejecuta por lo tanto de manera simultánea mediante cada uno de los procesadores virtuales 84, proporciona una operación de todo el sistema. En algunos casos, la tarea de indicación de funcionamiento final para el ciclo en solitario ejecuta la tarea enganchada. Esta técnica puede ser útil para escalar memoria, realizar operaciones de escritura y lectura de punto de comprobación de base de datos u otras tareas periódicas tales como registro diario de base de datos, registro y archivo. Las tareas pueden esperar y señalizar unas a las otras usando, por ejemplo, monitores, memoria compartida o semáforos.

Algunas tareas en las listas de ejecución 104 no relacionan datos en las cachés 92 (u otra partición de memoria) y por lo tanto pueden ejecutarse en cualquiera de los procesadores virtuales 84. Tales tareas pueden incluir una bandera en la estructura de tarea que indica que la tarea es movible. Las tareas pueden auto-modificar la bandera después de cada etapa para indicar si debe ejecutarse una siguiente etapa para la tarea en uno particular de los procesadores virtuales 84.

Para mejorar la utilización y reducir la congestión de los procesadores virtuales 84, las tareas se auto-equilibran para distribuir más equitativamente un número de tareas para cada una de las listas de ejecución 104 y, por lo tanto, para cada uno de los procesadores virtuales 84. En algunos casos, después de realizar cada etapa de una tarea, una tarea determina la longitud de la respectiva lista de ejecución 104 y las longitudes de listas de ejecución vecinas 104. Por ejemplo, una tarea puede determinar una longitud de (es decir, un número de tareas almacenadas mediante) la lista de ejecución 104B y las longitudes de las listas de ejecución 104A, 104C después de ejecutar una etapa de una tarea. Si la tarea determina que la longitud de la lista de ejecución 104B supera la longitud de cualquiera de las listas de ejecución 104A, 104C en un valor umbral, la tarea se migra a sí misma, si es movible, a la más corta de las listas de ejecución 104A, 104C. En algunos casos, las tareas tienen en cuenta incluso vecinas más remotas, es decir, no solamente vecinas más cercanas, cuando se realiza el re-equilibrio. De esta manera, las tareas se auto-organizan de manera autónoma de una manera equilibrada migrándose a sí mismas hacia procesadores virtuales 84 ligeramente cargados (y correspondientes núcleos). Las tareas pueden determinar longitudes de listas de ejecución vecinas, por ejemplo, intercambiando las longitudes en mensaje o leyendo un valor de memoria compartida.

La Figura 6 es un diagrama de bloques que ilustra el planificador 100A invocando múltiples tareas 120A-120K ("tareas 120") de la lista de ejecución 104A para realizar multiplexación por división en el tiempo de tareas de acuerdo con las técnicas descritas en esta divulgación. La lista de ejecución 104A en este ejemplo se implementa e ilustra como una lista vinculada circular. Por consiguiente, la tarea 120A que es la tarea de cabecera de la lista de ejecución 104A le sucede la tarea 120B. La tarea 120K es la tarea de cola de la lista de ejecución 104A y le sucede la tarea 120A. Cada una de las tareas 120 incluye una lista de una o más etapas para ejecución. Por ejemplo, la tarea 120A enumera las etapas $120A_1$-$120A_5$. Una etapa es un bloque auto-contenido de una o más instrucciones, tal como una función o referencia de función, para ejecución mediante el procesador virtual 84. Una etapa puede

invocar, como un elemento de ejecución, otras funciones definidas mediante programas que se ejecutan en el sistema.

Las etapas de las tareas 120 pueden incluir números variables y tipos de instrucciones y por lo tanto tienen diferentes longitudes de ejecución. En otras palabras, el tiempo requerido para ejecutar cada una de las etapas de las tareas 120 puede diferir de etapa a etapa. Las etapas de las tareas 120 se ejecutan atómicamente, es decir, desde la primera instrucción de la etapa hasta la última instrucción de la etapa sin interrupción. Después de completar una etapa de una de las tareas 120 en la lista de ejecución 104A, el planificador 100A invoca la siguiente etapa para la siguiente de las tareas 120 en la lista de ejecución. De esta manera, el planificador 100A que invoca unas diferentes de las tareas 120 realiza multiplexación por división en el tiempo “segmentando en etapas” las tareas. Es decir, en contradicción a “segmentar en tiempo” las tareas 120 de manera que a cada tarea se proporciona un corto periodo de tiempo por el núcleo durante el que la tarea puede ejecutar hasta reemplazarse, cada una de las tareas 120 continúa ejecutándose hasta que la tarea ha completado una etapa. Segmentar en etapas por lo tanto asegura la atomicidad de las etapas de las tareas 120.

Cada una de las tareas 120 mantiene un puntero de memoria, índice de etapa u otra referencia a la siguiente etapa para ejecución en la tarea asociada. Cuando el planificador 100A invoca una tarea, la tarea ejecuta la siguiente etapa y a continuación se pone en espera para devolver el control al planificador 100A, que invoca la siguiente tarea en la lista de ejecución 104A. Por ejemplo, la tarea 120K ejecuta la etapa $120C_2$ de la tarea 120K y a continuación devuelve el control al planificador 100A, que invoca la tarea 120A. La tarea 120A a continuación ejecuta la etapa $120A_3$. En algunos casos, un único subproceso de ejecución ejecuta cada una de las tareas 120 usando las técnicas de segmentación en etapas anteriormente descritas. Sin embargo, el único subproceso de ejecución puede ponerse en espera después de cada etapa, o después de ejecutar una etapa para la tarea de cola 120K de la lista de ejecución, por ejemplo, para permitir que se ejecuten subprocesos para tareas no de la lista de ejecución.

Los planificadores detallados 100 posibilitan que los procesadores virtuales 84 ejecuten múltiples transacciones de complejidad y duración variables. En general, las transacciones pueden caracterizarse como modificar una base de datos (por ejemplo, las sentencias de SQL INSERT, DELETE y UPDATE) o como consultar la base de datos (por ejemplo, una sentencia de SQL SELECT). Estas transacciones pueden caracterizarse además de acuerdo con su duración de ejecución. Por ejemplo, una transición que actualiza una única fila puede considerarse una transacción de ejecución corta, mientras una transacción que consulta toda la base de datos y/o realiza cálculos complejos/extendidos puede considerarse una transacción de ejecución larga. Como un ejemplo adicional más, una transacción de consulta basándose en SELECT FUTURE (descrito a continuación en mayor detalle) puede considerarse una transacción de ejecución de manera perpetua o “continua”. Los planificadores 100 pueden permitir intercalar la ejecución, mediante los procesadores virtuales 84, de diversas combinaciones de transacciones de ejecución, cortas, largas y de manera continua. En combinación con la capacidad de escalar más allá de los límites de memoria física y a un número arbitrario de núcleos, las técnicas pueden soportar consultas ricas y complejas en mezclas de carga de trabajo que incluyen transacciones de duración de ejecución variable, particularmente en el contexto de grandes números de transacciones recibidas desde un gran número de conexiones de cliente.

Las Figuras 7A-7B son diagramas de bloques que ilustran tareas de ejemplo que soportan la operación paralelizada de los aceleradores de datos transaccionales 13 de la Figura 1. Aunque se ilustran tareas como que pertenecen a grupos separados, por ejemplo, las tareas de red 140, cada una de las tareas opera sustancialmente independiente de otra como se describe en el presente documento.

Las tareas de red 140 soportan hacer de interfaz con clientes y posibilitan además la comunicación entre múltiples máquinas de agrupación que cooperan para implementar uno o más aceleradores 13. En este ejemplo, los conectores son la principal interfaz de comunicación entre máquinas de agrupación y entre un acelerador de datos transaccionales y uno o más clientes. Una instancia de la tarea de respuesta de conector 140A que se ejecuta en una máquina independiente o una máquina de agrupación escucha solicitudes de conexión de conector emitidas por clientes al sistema. Tras recibir una solicitud de conexión de conector, la tarea de respuesta de conector 140A produce nuevas instancias de la tarea de lectura de conector 140B y de la tarea de escritura de conector 140C específicas para la solicitud de conexión de conector. La nueva tarea de lectura de conector 140B y la tarea de escritura de conector 140C cooperan para completar la toma de contacto de conexión de conector y establecer una conexión de conector. La nueva tarea de lectura de conector 140B escucha solicitudes de servicio desde el cliente correspondiente. De esta manera, las tareas individuales que pueden ejecutarse en paralelo mediante múltiples núcleos implementan múltiples puntos de conexión paralelos con el sistema. Las técnicas pueden posibilitar por lo tanto que un único sistema maneje cientos de miles de conexiones concurrentes.

La tarea de finalización asíncrona 140D soporta tareas de lectura de conector 140B y tareas de escritura de conector 140C posibilitando operaciones de envío y recepción de conector asíncronas y facilitando la entrada/salida (E/S) de solicitud/respuesta de cliente de alto rendimiento. Un sistema puede producir una nueva tarea de finalización asíncrona 140D para cada conexión de conector. Las máquinas de agrupación pueden hacer de interfaz entre sí usando las tareas de red 140. La tarea de interconexión de anfitrión 140F gestiona conexiones de conector entre instancias de procesador virtual en dos o más máquinas de agrupación de una agrupación. Una instancia de la tarea de interconexión de anfitrión 140F se ejecuta en cada una de las máquinas de agrupación para establecer

conexiones de conector entre instancias de procesador virtual. La tarea de interconexión de anfitrión 140F puede, por ejemplo, crear una malla completa de conectores continuamente conectados entre todos los procesadores virtuales de las agrupaciones que residen en máquinas de agrupación separadas. Como alternativa, la tarea de interconexión de anfitrión 140F puede establecer conexiones entre tales procesadores virtuales según sea necesario para ejecutar solicitudes de clientes y facilitar la eficacia del sistema. Para establecer una nueva conexión de conector, la tarea de interconexión de anfitrión 140F, en este ejemplo, produce una nueva instancia de la tarea de conexión de conector 140E para la nueva conexión de conector, que a su vez produce nuevas instancias de la tarea de escritura de conector 140C y de la tarea de lectura de conector 140B.

Las tareas de ejecución de sentencias 142 incluyen tareas que representan realizaciones de ejemplo de las tareas anteriormente descritas con respecto a la Figura 3. Específicamente, la tarea de compilador 142A, la tarea de ejecución de sentencia 142B y la tarea de ejecución subordinada 142 pueden representar realizaciones de ejemplo de la tarea de compilador 44, la tarea de ejecución 46 y cualquiera de las tareas de ejecución subordinadas 48, respectivamente.

La tarea de arranque de sistema 146A inicializa un sistema de acuerdo con parámetros configurables y gestiona cargar al menos una porción de la base de datos relacional en memoria 27 desde el almacenamiento persistente. La tarea de cierre de sistema 146B almacena datos de sistema, incluyendo datos configurados durante la operación del sistema, a almacenamiento persistente para restauración posterior. Además, la tarea de cierre de sistema 146B puede gestionar la escritura al menos una porción de la base de datos relacional en memoria 27 al almacenamiento persistente.

La tarea de sentencia periódica 148A puede configurarse para ejecutar periódicamente una operación. Por ejemplo, una instancia de la tarea de sentencia periódica 148A puede configurarse para borrar periódicamente, desde una tabla de registro, sentencias previamente ejecutadas grabadas para facilitar compatibilidad con ACID. Este ejemplo es una forma de mantenimiento que agiliza el sistema eliminando datos superfluos. La tarea definida por el usuario 148B puede configurarse con instrucciones de usuario para ejecutar aplicaciones de usuario personalizadas con respecto a la base de datos relacional en memoria 27. De esta manera, los clientes tienen acceso al modelo de ejecución interna del sistema, y las técnicas de esta divulgación proporcionan un sistema altamente extensible al que los clientes pueden añadir tareas personalizadas. El modelo de tareas desvelado en el presente documento posibilita a los clientes y desarrolladores aumentar de manera incremental la sofisticación del sistema añadiendo simplemente tareas adicionales.

El motor de paralelización 22 proporciona una gama de servicios internos. Este incluye gestión de sesión, gestión de transacción, control de esquema, contenedores paralelizados, bloqueo, análisis, gestión de errores y generación de código máquina dinámico. Estos pueden accederse mediante un kit de herramientas u otra interfaz de programación de aplicación (API) para modificar la operación del motor de paralelización 22.

Las tareas de mantenimiento 144 administran recursos y administran el sistema. La tarea de recogida de residuos 144D realiza la recogida de residuos para recuperar memoria ocupada por objetos que ya no se hacen referencia por ningún proceso en el sistema. La tarea de recogida de residuos 144D es responsable de eliminar finalmente datos de campos de fila de estructuras (por ejemplo, índices) de la base de datos relacional en memoria 27 y recuperar la memoria. La tarea de ejecución de sentencia 142B elimina lógicamente una fila desde la base de datos en respuesta a sentencias de borrado entrantes. Sin embargo, una vez que una fila se ha marcado como borrada lógicamente, la tarea de ejecución de sentencia 142B inserta un puntero a la fila borrada en una lista de filas a eliminar/recuperar mediante la tarea de recogida de residuos 144D. Una tarea o tareas de recogida de residuos 144D aplican el algoritmo de asignación a cada fila para cada índice de la base de datos relacional en memoria 27 que hace referencia a la fila. La tarea o tareas de recogida de residuos 144D eliminan la fila de cada índice en que está y a continuación borran la estructura de fila, recuperando de esta manera la memoria que ocupaba la fila.

La tarea de indicación de funcionamiento 144B y la tarea de alarma 144E pueden representar una realización de ejemplo de las tareas de indicación de funcionamiento 102 de la Figura 5. Las instancias de la tarea de alarma 144E gestionan cada una un temporizador hardware y una lista de espera de un procesador virtual de acuerdo con las técnicas descritas con respecto a la Figura 5. La tarea de licencia 144C asegura que el sistema está operando con una licencia válida. La tarea de estadística 144F mide el rendimiento y otras métricas del sistema y comunica las estadísticas mediante una instancia de tarea de escritura de conector 140C a una entidad de gestión. Por ejemplo, una instancia de tarea de estadística 144F puede cronometrar etapas ejecutadas por subprocesos, monitorizar el número de tareas en el sistema, monitorizar el rendimiento de solicitud de cliente o tiempo de respuesta, y monitorizar una tasa de llegada de solicitud de cliente. La tarea de monitor 144A comprueba periódicamente el estado de todas las otras tareas en el sistema para informar errores/advertencias y para facilitar la corrección/manejo de errores.

Las tareas de disco 150 proporcionan compatibilidad de durabilidad para la base de datos relacional en memoria 27. La tarea de escritura de registro diario 150A escribe el estado para sentencias ejecutadas al registro diario de transacción 152, un dispositivo de almacenamiento legible por ordenador. Tras un fallo de transacción u otro fallo operacional relacionado con la base de datos, la tarea de lectura de registro diario 150C lee el estado escrito para

las sentencias anteriormente ejecutadas, y la tarea de restauración de registro diario 150B restaura, si fuera necesario, el estado a la memoria para restaurar la base de datos relacional en memoria 27 a un estado conocido. Una instancia de la tarea de sentencia periódica 148A puede determinar periódicamente entradas de registro diario obsoletas y borrar tales entradas de un registro diario de transacción 152.

Las tareas relacionadas con puntos de comprobación persisten y restauran porciones de la base de datos relacional en memoria 27 a/desde el punto de comprobación de sistema 154, un dispositivo de almacenamiento legible por ordenador. La tarea de toma de punto de comprobación 150D determina una porción de la memoria para escribir en disco y dirige la tarea de escritura de punto de comprobación 150E para escribir la porción como un punto de comprobación en disco. En el caso de un fallo relacionado con la base de datos, la tarea de restauración de punto de comprobación 150F determina una porción o porciones de memoria para restaurar desde puntos de comprobación anteriormente escritos y dirige la tarea de lectura de punto de comprobación 150G para leer los puntos de comprobación y reinsertar los datos de punto de comprobación a localizaciones de memoria apropiadas. Esto posibilita a la tarea de lectura de registro diario 150C leer y almacenar únicamente aquellas transacciones aplicadas después de la tarea de toma de punto de comprobación 150D almacenada en el punto de comprobación al punto de comprobación de sistema 154.

La tarea de escritura de página 150H y la tarea de lectura de página 150I paginan datos en memoria a almacenamiento secundario representado mediante el almacén de datos 156, un dispositivo de almacenamiento legible por ordenador, para escalar memoria utilizada mediante la base de datos relacional en memoria 27. La tarea de escritura de página 150H identifica elementos caducados (por ejemplo, filas) de objetos de base de datos en la base de datos relacional en memoria 27 y, tras identificar elementos caducados, escribe datos para los elementos caducados en el almacén de datos store 156. Además, la tarea de escritura de página 150H borra posteriormente los elementos caducados. Cuando una tarea que se ejecuta en el sistema requiere acceso a elementos escritos en el almacén de datos 156, la tarea de lectura de página 1501 lee los elementos del almacén de datos e inserta los datos para los elementos, usando transacciones, a la base de datos relacional en memoria 27.

La tarea de escritura de registro 150J registra operaciones de sistema al registro de sistema 158, un dispositivo de almacenamiento legible por ordenador. Las tareas de archivo 150K identifican entradas de registro diario y/o puntos de comprobación que se hacen obsoletos por puntos de comprobación posteriores y escriben los datos a almacenamiento terciario representado mediante el archivo 160, un dispositivo de almacenamiento legible por ordenador.

La Figura 8 es un diagrama de bloques que ilustra un acelerador de datos transaccionales 178 de ejemplo que escribe a múltiples objetos de datos en paralelo de acuerdo con las técnicas descritas en el presente documento. El acelerador de datos transaccionales 178 incluye los núcleos 185A-185D (“núcleos 185”). Los índices 186, 188 son cada uno un índice para un campo diferente de un objeto de tabla (no mostrado) de la base de datos relacional en memoria 27. Por ejemplo, el objeto de tabla puede incluir dos campos, ID_EMPLEADO y NOMBRE_EMPLEADO cada uno indexado mediante uno de los índices separados 186, 188. Cada uno de los índices parciales 186A-186D divide y almacena datos parciales para el índice 186 de acuerdo con un algoritmo de asignación. De manera similar, cada uno de los índices parciales 1188A-188D divide y almacena datos parciales para el índice 188 de acuerdo con el algoritmo de asignación.

El cliente 180 emite al acelerador de datos transaccionales 178 una sentencia de solicitud que hace referencia a ambos campos en los que se indexan los índices 186, 188. Por ejemplo, la sentencia de solicitud puede ser una sentencia de SQL INSERT, DELETE o UPDATE para respectivamente insertar, borrar o actualizar una fila en/del objeto de tabla en la que están basados los índices 186, 188. Como otro ejemplo, la sentencia de solicitud puede ser una sentencia de SQL SELECT para obtener todas las filas que coinciden con el criterio que hace referencia a ambos campos en los que se indexan los índices 186,188. Por lo tanto, las técnicas pueden permitir la paralelización de muchos tipos diferentes de operaciones de lenguaje declarativo (por ejemplo, SQL) para no únicamente consultar sino también para modificar una base de datos en memoria.

La tarea de ejecución de sentencia 182 recibe, mediante una conexión de conector y una tarea de compilador (ninguna mostrada en la Figura 8), transacciones 181 que forman un plan de ejecución para ejecutar la sentencia de solicitud desde el cliente 180. Las transacciones 181 incluyen una primera sub-transacción para el índice 186 y una segunda sub-transacción para el índice 188. La tarea de ejecución de sentencia 182 produce las tareas de ejecución subordinadas 183, 184 para ejecutar la primera y segunda sub-transacciones de las transacciones 181.

Por ejemplo, en el caso del ejemplo de la sentencia de SQL INSERT, anteriormente, la tarea de ejecución de sentencia 182 crea en primer lugar y añade la nueva fila al objeto de tabla de acuerdo con datos de fila recibidos en la sentencia de solicitud. La tarea de ejecución de sentencia 182 a continuación realiza un algoritmo de asignación usando el valor de campo de la fila para el campo en el que está basado el índice 186 y, basándose en la salida del algoritmo de asignación, asigna el valor de campo al núcleo 185B. La tarea de ejecución de sentencia 182 produce la tarea de ejecución subordinada 183 al núcleo 185B y dirige la tarea producida para insertar una fila de índice para los nuevos datos al índice parcial 186B. La tarea de ejecución subordinada 183 añade la fila de índice al índice

parcial 186B con un puntero de memoria a la nueva fila añadida a la tarea de ejecución de sentencia 182 al objeto de tabla.

Además, la tarea de ejecución de sentencia 182 realiza un algoritmo de asignación usando el valor de campo de la fila para el campo en que está basado el índice 188 y, basándose en la salida del algoritmo de asignación, asigna el valor de campo al núcleo 185D. La tarea de ejecución de sentencia 182 produce la tarea de ejecución subordinada 184 al núcleo 185D y dirige la tarea producida para insertar una fila de índice para el nuevo dato al índice parcial 188D. La tarea de ejecución subordinada 184 añade la fila de índice al índice parcial 188D con un puntero de memoria a la nueva fila añadida a la tarea de ejecución de sentencia 182 al objeto de tabla. De esta manera, las tareas de ejecución subordinadas 183, 184 pueden ejecutarse concurrentemente y la inserción de nuevas filas de índice a los índices 186, 188 puede tener lugar en paralelo, en lugar de en serie. En algunos casos, las transacciones 181 pueden incluir sub-transacciones que producen cada una tareas para escribir a los índices parciales 186. Por ejemplo, las transacciones 181 pueden incluir sub-transacciones para escribir a los respectivos índices parciales 186B, 186D. Sin embargo, las tareas de ejecución subordinadas 183, 184 pueden ejecutarse concurrentemente para modificar simultáneamente los índices parciales 186B, 186D para el mismo objeto de base de datos, es decir, el índice 186. Las tareas de ejecución subordinadas 183, 184 devuelven datos y/o información de estado a la tarea de ejecución de sentencia 182, que devuelve un resultado 189 al cliente 180 mediante una tarea de conexión de conector (no mostrada en la Figura 8).

La Figura 9 es un diagrama de bloques que ilustra un acelerador de datos transaccionales 190 de ejemplo que incluye máquinas de agrupación 196A-196C (“máquinas de agrupación 196”) que presentan una interfaz unificada para un cliente y escriben en estructuras de datos parciales de acuerdo con las técnicas descritas en el presente documento. Cada una de las máquinas de agrupación 196 incluye un núcleo 1, núcleo 2 y núcleo 3. Por ejemplo, la máquina de agrupación 196A incluye el núcleo $196A_1$, $196A_2$ y $196A_3$. Las máquinas de agrupación 196A, 196B ejecutan respectivas tareas de conexión de conectores 195A, 195B con las que hacer de interfaz para comunicar datos e información de tareas. Cada núcleo en las máquinas 196 del acelerador de datos transaccionales 190 incluye una estructura de datos parciales para el objeto de datos 194 de la base de datos relacional en memoria 27. Por ejemplo, el núcleo $196A_1$ incluye la estructura parcial $194A_1$ y el núcleo $196B_2$ incluye la estructura parcial $194B_2$.

El cliente 191 emite al acelerador de datos transaccionales 190 una sentencia de solicitud que hace referencia a datos para el objeto de datos 194. La tarea de ejecución de sentencia 193 recibe, mediante una tarea de compilador (no mostrado), la transacción 192 que forma un plan de ejecución para ejecutar la sentencia de solicitud desde el cliente 191. La tarea de ejecución de sentencia 193 realiza un algoritmo de asignación agrupado, tal como una función de troceo de agrupación, usando los datos para la transacción 192. El algoritmo de asignación agrupado emite dos índices, un primer índice en la dimensión de máquina y un segundo índice en la dimensión de núcleo. La tarea de ejecución de sentencia 193 de esta manera usa el algoritmo de asignación agrupado para identificar de manera determinística un núcleo apropiado de las máquinas 196 para ejecutar la transacción 192.

En el ejemplo ilustrado, el algoritmo de asignación agrupado emite el índice de máquina 2 y el núcleo 2 para indicar que una tarea que opera en el núcleo $196B_2$ debería ejecuta la transacción 192. Puesto que tarea de ejecución de sentencia 193 se ejecuta en la máquina 196A que no es la misma que la máquina 196B para el núcleo $196B_2$, la tarea de ejecución de sentencia 193 establece una conexión de conector entre las máquinas 196A, 196B mediante respectivas tareas de conexión de conector 195A, 195B. La tarea de ejecución de sentencia 193 a continuación produce la tarea de ejecución subordinada 196 usando las tareas de conexión de conector 195, y la tarea de ejecución subordinada 196 ejecuta la transacción 192 en la estructura parcial $194B_2$ asociada con el núcleo $196B_2$. En algunos casos, la tarea de ejecución subordinada 196 puede devolver un resultado de transacción 192 a la tarea de ejecución de sentencia 193 mediante las tareas de conexión de conector 195. La tarea de ejecución de sentencia 193 puede producir la tarea de ejecución subordinada 196 en la máquina 196B mediante, por ejemplo, poner en serie y enviar las etapas de la tarea mediante las tareas de conexión de conector 195. Las tareas de conexión de conector 195 por lo tanto actúan en este caso como un intermediario para la tarea de ejecución de sentencia 193.

La Figura 10 ilustra un sistema de ejemplo 200 en que se usan las instancias del acelerador 13 como bloques de creación para formar una estructura en niveles para enviar iterativamente datos seleccionados a un gran número de clientes individuales (usuarios) 201 y crear sesiones de difusión privadas interactivas (canales) en un flujo de datos de difusión en masa homogéneo.

En el ejemplo de la Figura 10, el sistema 200 incluye una pluralidad de aceleradores de datos transaccionales 13 dispuestos en una estructura de tres niveles que tiene el nivel de núcleo 202, el nivel de distribución 204 y el nivel de extremo 206. Los consumidores de datos 16, que pueden ser en el orden de millones de dispositivos, establecen cada uno una única consulta 207 con los aceleradores de datos transaccionales 13 del nivel de extremo 206. A su vez, los aceleradores de datos transaccionales 13 del nivel de extremo 206 establecen cada uno consultas 209 con los aceleradores de datos transaccionales 13 del nivel de distribución 204. Es decir, los motores de introducción de datos 26 en los aceleradores de datos transaccionales 13 del nivel de extremo 206 establecen conexiones con los motores de entrega de contenido 30 de los aceleradores de datos transaccionales 13 en el nivel de distribución 204 y proporcionan consultas agregadas 209 a los motores de entrega de contenido, donde las consultas agregadas 209 son cada una un ejemplo de una transacción agregada. Es decir, cada motor de entrega de contenido 30 en el nivel

de extremo 206 de los aceleradores de datos transaccionales 13 calcula una consulta agregada 209 que representa todos los datos especificados de las consultas específicas de cliente recibidas desde los consumidores de datos 16. En otras palabras, la consulta agregada 209 calculada mediante cada motor de entrega de contenido 30 en el nivel de distribución 204 especifica una pluralidad de ajustes de condición que corresponden a la condición especificada por los consumidores de datos 16 con los que el motor de entrega de contenido ha establecido conexiones.

En una realización de ejemplo, el motor de aceleración de base de datos 30 de cada acelerador 13 presenta la API basada en SQL que se ha mejorado para permitir a los consumidores de datos 16 especificar fácilmente consultas continuas. Por ejemplo, en una realización la API basada en SQL soporta un testigo opcional *future* para incluirse en cualquier sentencia *select* emitida para indicar que la consulta definida mediante la sentencia *select* se ha de aplicar continuamente a nuevos datos aún no recibidos. Por ejemplo, un primer consumidor de datos 16 puede emitir una consulta como sigue:

SELECT FUTURE stock_price, daily_volume FROM stock_table WHERE stock_symbol = 'IBM'

y un segundo consumidor de datos puede emitir una consulta como sigue:

SELECT FUTURE stock_price, daily_high FROM stock_table WHERE stock_symbol = 'GE"

En este caso, ambas consultas incluyen la nueva palabra clave que provoca que el motor de aceleración de base de datos de recepción 30 indique que trate la consulta como una consulta continua en lugar de una consulta de una vez. En este caso, una consulta agregada puede calcularse desde las consultas específicas de cliente como:

SELECT FUTURE stock_price, *daily_volume, daily_high FROM stock_table WHERE stock_symbol = 'GE' or stock_symbol= 'IBM'.*

A su vez, los motores de introducción de datos 26 de los aceleradores de datos transaccionales 13 del nivel de distribución 204 establecen conexiones y proporcionan consultas agregadas 211 a los motores de entrega de contenidos 30 del acelerador 13 en el nivel de núcleo 202. El nivel de núcleo 202 representa una agrupación de uno o más aceleradores de datos transaccionales 13 que operan en un flujo de datos de transacción, como se ha descrito anteriormente, desde una o más fuentes. Si tiene lugar un cambio de datos, los datos actualizados se envían automáticamente desde el nivel de núcleo 202 a aquellos consumidores de datos 16 para los que los datos actualizados coinciden con las condiciones definidas mediante la consulta del cliente 207. En cada nivel, los motores de entrega de contenido 30 distribuyen los datos a los motores de introducción de datos 26 para alimentación de la base de datos en memoria masivamente paralela como se describe en el presente documento hasta que la actualizada se envía a los consumidores de datos 16. Las técnicas de paralelización descritas en el presente documento permiten que este proceso sea extremadamente rápido. Por ejemplo, pueden soportarse millones de consumidores de datos 16 usando el ejemplo de estructura de tres niveles del sistema 200 de manera que puede enviarse continuamente datos cambiantes en el nivel de núcleo 202 a los consumidores de datos 16 en el orden de aproximadamente un milisegundo. Esto permite que los datos procesados en el nivel de núcleo 202 se 'envíen' (entreguen) desde la base de datos en memoria como un flujo de datos dirigido a los consumidores de datos 16. Aunque se ha descrito con respecto a tres niveles, pueden usarse otros niveles. Por ejemplo, el ejemplo con cuatro niveles, los datos podrían enviarse a billones de consumidores de datos de una manera rentable y oportuna.

La Figura 11 ilustra un sistema de ejemplo 250 en que se usan instancias del acelerador 13 como bloques de creación para formar una estructura en niveles para recibir datos desde un gran número de fuentes de datos individuales (usuarios) 251. Las fuentes de datos 12 pueden ser dispositivos informáticos de usuario final (por ejemplo, dispositivos móviles), sensores físicos (por ejemplo, sensores acústicos para recoger datos desde ondas de choque o sensores de control de tráfico para proporcionar datos en tiempo real de tráfico que se mueve a través de un gran área metropolitana) o cualquier dispositivo que produce datos. El sistema 250 proporciona una manera elegante y rentable para consumir y procesar datos desde grandes números de productores de datos.

En el ejemplo de la Figura 11, el sistema 250 incluye una pluralidad de aceleradores de datos transaccionales 13 dispuestos en una estructura en tres niveles que tiene el nivel de núcleo 252, el nivel de distribución de entrada 254 y el nivel de extremo 256. Las fuentes de datos 12, que pueden ser en el orden de millones de dispositivos, establecen cada una conexiones con y envían datos transaccionales a los aceleradores de datos transaccionales 13 del nivel de extremo 256. A su vez, los aceleradores de datos transaccionales 13 del nivel de extremo 256 establecen cada uno conexiones con y envían datos a los aceleradores de datos transaccionales 13 del nivel de distribución de entrada 254. Es decir, los motores de entrega de contenido 30 en los aceleradores de datos transaccionales 13 del nivel de extremo 206 establecen conexiones con los motores de introducción de datos 26 del acelerador 13 en el nivel de distribución de entrada 254 y proporcionan datos a los motores de introducción de datos 26. A su vez, los motores de entrega de contenido 30 de los aceleradores de datos transaccionales 13 del nivel de distribución de entrada 254 establecen conexiones y proporcionan datos a los motores de introducción de datos 26 de los aceleradores de datos transaccionales 13 en el nivel de núcleo 252. El nivel de núcleo 252 representa, una agrupación de uno o más aceleradores de datos transaccionales 13 que operan en datos de transacción como se ha descrito anteriormente. El nivel de núcleo 252 puede procesar y emitir los datos recibidos a almacenes de datos o

clientes interesados que usan canales de difusión privados como se describe con respecto al sistema 200 de la Figura 10.

La Figura 12 es un diagrama de bloques que ilustra un sistema de ejemplo que utiliza las técnicas descritas en el presente documento para proporcionar aceleración de datos transaccionales de alto rendimiento. Los caracteres de referencia similares se usan para indicar elementos similares de la Figura 1. En el sistema de ejemplo 300 de la Figura 12, como en el sistema de ejemplo 10 de la Figura 1, un conjunto de aceleradores de datos transaccionales 13A-13N (colectivamente, “aceleradores de datos transaccionales 13”) se organizan para formar una agrupación 11 de aceleradores de datos. Los aceleradores de datos transaccionales operan para introducir continuamente y procesar grandes cantidades de transacciones de datos desde las fuentes de datos 12 y entregar los datos procesados a las fuentes de datos 16. Las fuentes de datos 12 y el consumidor de datos 16 pueden ser cualquiera que origine o consuma datos, tales como sistemas, aplicaciones o bases de datos.

En este ejemplo, los consumidores de datos 16 responden para difundir datos entregados mediante los aceleradores de datos transaccionales 13. Las respuestas a datos de difusión desde los consumidores de datos 16 representan datos para las fuentes de datos 12, que se introducen mediante los aceleradores de datos transaccionales 13, se procesan y usan para entregar datos refinados, agregados procesados de otra manera a los consumidores de datos 16. Aunque las operaciones de los aceleradores de datos transaccionales 13 pueden ser sustancialmente similares en tanto el sistema 10 de la Figura 1 como el sistema 300 de la Figura 12, el sistema 300 implica un “bucle de realimentación” de datos de difusión en tiempo real (o cercano a tiempo real) y respuestas de datos de difusión y por lo tanto ilustra que las técnicas de esta divulgación, como se implementan mediante los aceleradores de datos transaccionales 13, pueden aplicarse para acelerar la entrega interactiva de resultados de consulta activados por usuario y otras aplicaciones interactivas.

Las técnicas descritas en esta divulgación pueden implementarse, al menos en parte, en hardware, software, firmware o cualquier combinación de los mismos. Por ejemplo, diversos aspectos de las técnicas descritas pueden implementarse en uno o más procesadores, incluyendo uno o más microprocesadores, procesadores de señales digitales (DSP), circuitos integrados específicos de la aplicación (ASIC), campos de matrices de puertas programables (FPGA) o cualquier otra circuitería lógica integrada o discreta equivalente, así como cualquier combinación de tales componentes. El término “procesador” o “circuitería de procesamiento” puede referirse en general a cualquiera de la circuitería lógica anterior, en solitario o en combinación con otra circuitería lógica, o cualquier otra circuitería equivalente. Una unidad de control que comprende hardware puede realizar también una o más de las técnicas de esta divulgación.

Tal hardware, software y firmware pueden implementarse en el mismo dispositivo o en dispositivos separados para soportar las diversas operaciones y funciones descritas en esta divulgación. Además, cualquiera de las unidades descritas, módulos o componentes pueden implementarse juntos o por separado como dispositivos de lógica discreta pero interoperables. La representación de diferentes características como módulos o unidades se pretende para destacar diferentes aspectos funcionales y no implica necesariamente que tales módulos o unidades deben realizarse mediante hardware separado o componentes de software. En su lugar, la funcionalidad asociada con uno o más módulos o unidades puede realizarse mediante hardware separado o componentes de software, o integrarse en componentes de hardware o software comunes o separados.

Las técnicas descritas en esta divulgación pueden realizarse o codificarse también en un medio legible por ordenador, tal como un medio legible por ordenador no transitorio o medio o dispositivo de almacenamiento legible por ordenador que contiene instrucciones. Las instrucciones realizadas o codificadas en un medio legible por ordenador pueden provocar que un procesador programable, u otro procesador, realice el método, por ejemplo, cuando se ejecutan las instrucciones. El medio de almacenamiento legible por ordenador puede incluir memoria de acceso aleatorio (RAM), memoria de solo lectura (ROM), memoria de solo lectura programable (PROM), memoria de solo lectura programable borrable (EPROM), memoria de sólo lectura electrónicamente programable borrable (EEPROM), memoria flash, un disco duro, un CD-ROM, un disco flexible, un casete, medio magnético, medio óptico u otro medio de almacenamiento legible por ordenador. Debería entenderse que la expresión “medio de almacenamiento legible por ordenador” se refiere a medio de almacenamiento físico, y no señales u ondas portadora, aunque la expresión “medio legible por ordenador” puede incluir medio transitorio tal como señales, además de medio de almacenamiento físico.

En un ejemplo, un dispositivo comprende un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución y una pluralidad de máquinas virtuales que opera cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de los núcleos de ejecución. El dispositivo comprende también una base de datos en memoria que comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución. El dispositivo comprende también una pluralidad de tareas que se ejecutan en las máquinas virtuales para descomponer una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un método comprende ejecutar una pluralidad de máquinas virtuales que operan cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de una pluralidad de núcleos de ejecución de un procesador hardware multi-núcleo de un dispositivo informático. El método comprende también asociar a uno diferente de los núcleos de ejecución con cada una de una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria almacena datos para una base de datos en memoria. El método comprende también ejecutar una pluralidad de tareas con las máquinas virtuales, en el que la pluralidad de tareas descompone una transacción entrante para la base de datos en memoria en sub-transacciones, despliegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un dispositivo de almacenamiento legible por ordenador que comprende instrucciones que, cuando se ejecutan, provocan que un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución ejecute una pluralidad de máquinas virtuales que opera cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de la pluralidad de núcleos de ejecución, asociar uno diferente de los núcleos de ejecución con cada una de una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria almacena datos para una base de datos en memoria, y ejecutar una pluralidad de tareas con las máquinas virtuales, en el que la pluralidad de tareas descompone una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un dispositivo comprende un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución. El dispositivo comprende también una base de datos en memoria que comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución, y un motor de entrega de contenido que configura una pluralidad de canales de datos privados para entregar cada uno datos únicos a uno correspondiente de una pluralidad de consumidores de datos. El dispositivo comprende también un motor de paralelización que despliega una pluralidad de tareas para ejecutar concurrentemente en los núcleos para realizar concurrentemente transacciones en la base de datos en memoria en respuesta a consultas desde el motor de entrega de contenido.

En otro ejemplo, un método comprende configurar, con un motor de entrega de contenido, una pluralidad de canales de datos privados para entregar cada uno datos únicos a uno correspondiente de una pluralidad de consumidores de datos. El método comprende también desplegar, con un motor de paralelización que se ejecuta en un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución, una pluralidad de tareas para ejecutar concurrentemente en los núcleos para realizar concurrentemente transacciones en una base de datos en memoria en respuesta a consultas desde el motor de entrega de contenido, en el que la base de datos en memoria comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución.

En otro ejemplo, un dispositivo de almacenamiento legible por ordenador comprende instrucciones que, cuando se ejecutan, provocan que un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución configure, con un motor de entrega de contenido, una pluralidad de canales de datos privados para entregar cada uno datos únicos a uno correspondiente de una pluralidad de consumidores de datos, y desplegar, con un motor de paralelización, una pluralidad de tareas para ejecutar concurrentemente en los núcleos para realizar concurrentemente transacciones en una base de datos en memoria en respuesta a consultas desde el motor de entrega de contenido, en el que la base de datos en memoria comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución.

En otro ejemplo, un dispositivo comprende un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución y una base de datos en memoria que comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución. El dispositivo comprende también un motor de introducción de datos que establece una pluralidad de conexiones de red concurrentes a los dispositivos externos para recibir transacciones entrantes. El dispositivo comprende también un motor de paralelización que descompone transacciones entrantes para la base de datos en memoria en sub-transacciones, despliega las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecuta concurrentemente las sub-transacciones de las transacciones entrantes en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un método comprende establecer una pluralidad de conexiones de red concurrentes a dispositivos externos con un motor de introducción de datos, y recibir transacciones entrantes con el motor de introducción de

datos mediante las conexiones de red concurrentes. El método comprende también descomponer, con un motor de paralelización que se ejecuta en un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución, transacciones entrantes para una base de datos en memoria en sub-transacciones, en el que la base de datos en memoria comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución. El método comprende también desplegar, con el motor de paralelización, las sub-transacciones a los núcleos de ejecución asociados con respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de las transacciones entrantes en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

En otro ejemplo, un dispositivo de almacenamiento legible por ordenador comprende instrucciones que, cuando se ejecutan, provocan que un procesador hardware multi-núcleo que tiene una pluralidad de núcleos de ejecución establezcan una pluralidad de conexiones de red concurrentes a dispositivos externos con un motor de introducción de datos, recibir transacciones entrantes con el motor de introducción de datos mediante las conexiones de red concurrentes. Las instrucciones, cuando se ejecutan, producen también que el procesador hardware multi-núcleo descomponga, con un motor de paralelización, transacciones entrantes para una base de datos en memoria en sub-transacciones, en el que la base de datos en memoria comprende datos almacenados en una pluralidad de particiones de memoria, en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución, despliegue, con el motor de paralelización, las sub-transacciones a los núcleos de ejecución asociados con respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecute concurrentemente las sub-transacciones de las transacciones entrantes en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

REIVINDICACIONES

1. Un método que comprende:

ejecutar una pluralidad de máquinas virtuales (20) que operan cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de una pluralidad de núcleos de ejecución (52) de un procesador hardware multi-núcleo (24) de un dispositivo informático (13);
asociar uno diferente de los núcleos de ejecución con cada una de una pluralidad de particiones de memoria (51), en el que cada una de las particiones de memoria almacena datos para una base de datos en memoria (27); y
ejecutar una pluralidad de tareas con las máquinas virtuales, en el que la pluralidad de tareas descomponen una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

2. El método de la reivindicación 1, que comprende además:

aplicar, con una tarea de ejecución de sentencia en una de las máquinas virtuales, un algoritmo de asignación a las sub-transacciones para identificar los respectivos núcleos de ejecución asociados con las particiones que almacenan datos para las sub-transacciones,
en el que la tarea de ejecución de sentencia produce y despliega, para cada una de las sub-transacciones, una tarea de sub-ejecución para ejecutar la sub-transacción en el núcleo identificado de los núcleos de ejecución para la sub-transacción, en el que las tareas de sub-ejecución para las correspondientes sub-transacciones se ejecutan concurrentemente en las máquinas virtuales.

3. El método de la reivindicación 2,
en el que la tarea de ejecución de sentencia aplica el algoritmo de asignación a una pluralidad de datos referenciados mediante la transacción entrante,
en el que el algoritmo de asignación identifica uno diferente de los núcleos de ejecución para cada uno de la pluralidad de datos referenciados mediante la transacción entrante,
en el que cada una de las sub-transacciones comprende una transacción para uno correspondiente de la pluralidad de datos referenciados mediante la transacción entrante, y en el que la tarea de ejecución de sentencia produce una tarea de sub-ejecución al uno identificado de los núcleos de ejecución para cada uno de la pluralidad de datos para ejecutar la sub-transacción correspondiente.

4. El método de cualquiera de las reivindicaciones 2-3,
en el que la tarea de ejecución de sentencia aplica una gramática transaccional y un esquema a la transacción entrante para generar sub-transacciones que comprende cada una etapas de una biblioteca de etapas,
en el que el esquema que describe objetos de base de datos de la base de datos en memoria,
en el que la biblioteca de etapas comprende una o más etapas especificando cada una instrucciones ejecutables por la pluralidad de núcleos, y
en el que la gramática transaccional describe un lenguaje al que se ajusta la transacción entrante y comprende una o más reglas de sustitución que hacen referencia a etapas de la biblioteca de etapas.

5. El método de cualquiera de las reivindicaciones 1-4, en el que las máquinas virtuales ejecutan asíncronamente la pluralidad de tareas, comprendiendo el método además:

ejecutar una tarea de indicación de funcionamiento en cada una de las máquinas virtuales de acuerdo con un periodo definido para provocar, tras la ejecución de la tarea de indicación de funcionamiento, que la máquina virtual correspondiente espere para ejecutar cualquier tarea adicional hasta que la máquina virtual correspondiente reciba una señal.

6. El método de la reivindicación 5,
en el que la tarea de indicación de funcionamiento comprende un gancho de tarea que especifica una tarea enganchada,
en el que una de las máquinas virtuales ejecuta la tarea enganchada, y
en el que la una de las máquinas virtuales envía la señal a las otras máquinas virtuales después de finalizar la tarea enganchada.

7. El método de la reivindicación 6, en el que la tarea enganchada comprende una tarea de punto de comprobación que, cuando se ejecuta, genera y almacena un punto de comprobación para la base de datos en memoria.

8. El método de cualquiera de las reivindicaciones 1-7,
en el que cada una de las máquinas virtuales comprende un planificador para el núcleo de ejecución correspondiente, y en el que cada uno de los planificadores mantiene una estructura de datos de lista de ejecución

separada que almacena una referencia a tareas planificadas para ejecución mediante el uno correspondiente de la pluralidad de núcleos de ejecución.

9. El método de la reivindicación 8, en el que cada una de las máquinas virtuales incluye una estructura de datos de cola cruzada que almacena referencias a tareas migradas a la máquina virtual mediante un planificador de una diferente de las máquinas virtuales.

10. El método de la reivindicación 9, en el que un planificador de una primera de las máquinas virtuales pone en cola una referencia a una primera tarea a la estructura de datos de cola cruzada de una segunda de las máquinas virtuales mientras la segunda máquina virtual ejecuta simultáneamente una segunda tarea referenciada en la lista de ejecución mantenida mediante el planificador de la segunda máquina virtual.

11. El método de cualquiera de las reivindicaciones 1-10,
en el que cada una de las máquinas virtuales define una estructura de datos de lista de espera que almacena referencias a tareas en espera ordenadas mediante tiempo de espera restante para las tareas en espera, y
en el que cada uno de la pluralidad de núcleos de ejecución comprende un temporizador, comprendiendo el método además:

ejecutar una tarea de alarma en cada una de las máquinas virtuales para establecer un tiempo de expiración para el temporizador para el núcleo de ejecución correspondiente basándose al menos en el tiempo de espera restante más pequeño para la tarea en espera de la estructura de datos de lista de espera de la máquina virtual,
en el que la expiración del temporizador provoca que la tarea de alarma active la siguiente tarea en espera de la lista de espera.

12. El método de cualquiera de las reivindicaciones 1-11, que comprende además:

recibir, con un motor de entrega de contenido para el dispositivo informático, una primera transacción que solicita primeros datos y una segunda transacción que solicita segundos datos;
calcular, con el motor de entrega de contenido, una transacción agregada que solicita el primer dato y el segundo dato; y
emitir la transacción agregada a otro dispositivo.

13. El método de la reivindicación 12, que comprende además:

recibir, con el motor de entrega de contenido, una transacción agregada que solicita primeros datos y segundos datos;
dirigir el motor de paralelización para ejecutar una transacción agregada para agregar el primer dato y el segundo dato; y
emitir, con el motor de entrega de contenido, el primer dato y el segundo dato agregados mediante el motor de paralelización en respuesta a la transacción agregada.

14. Un dispositivo (13) que comprende:

un procesador hardware multi-núcleo (24) que tiene una pluralidad de núcleos de ejecución (52);
una pluralidad de máquinas virtuales (20) que operan cada una de acuerdo con un conjunto de instrucciones virtual, en el que cada una de las máquinas virtuales se ejecuta en uno diferente de los núcleos de ejecución;
una base de datos en memoria (27) que comprende datos almacenados en una pluralidad de particiones de memoria (51), en el que cada una de las particiones de memoria está asociada con uno diferente de los núcleos de ejecución; y
una pluralidad de las tareas que se ejecutan en las máquinas virtuales para descomponer una transacción entrante para la base de datos en memoria en sub-transacciones, desplegar las sub-transacciones a los núcleos de ejecución asociados con las respectivas particiones de memoria que almacenan datos para las sub-transacciones, y ejecutar concurrentemente las sub-transacciones de la transacción entrante en los respectivos núcleos de ejecución a los que se despliegan las sub-transacciones.

15. Un medio de almacenamiento legible por ordenador codificado con instrucciones que, cuando se ejecutan en un procesador hardware multi-núcleo (24), provocan que dicho procesador (24) realice el método de cualquiera de las reivindicaciones 1-13.

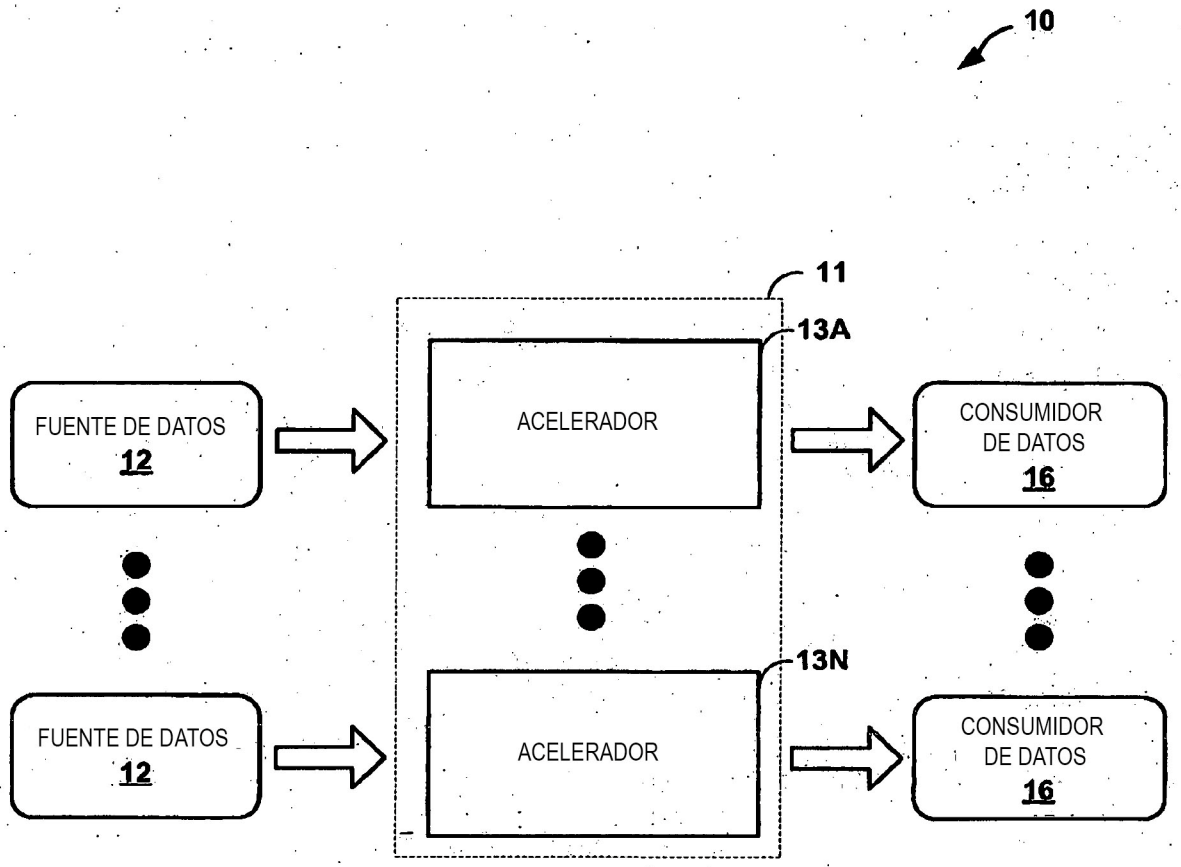
10
11
13A
FUENTE DE DATOS
12
ACELERADOR
CONSUMIDOR
DE DATOS
16
13N
FUENTE DE DATOS
12
ACELERADOR
CONSUMIDOR
DE DATOS
16

FIG. 1

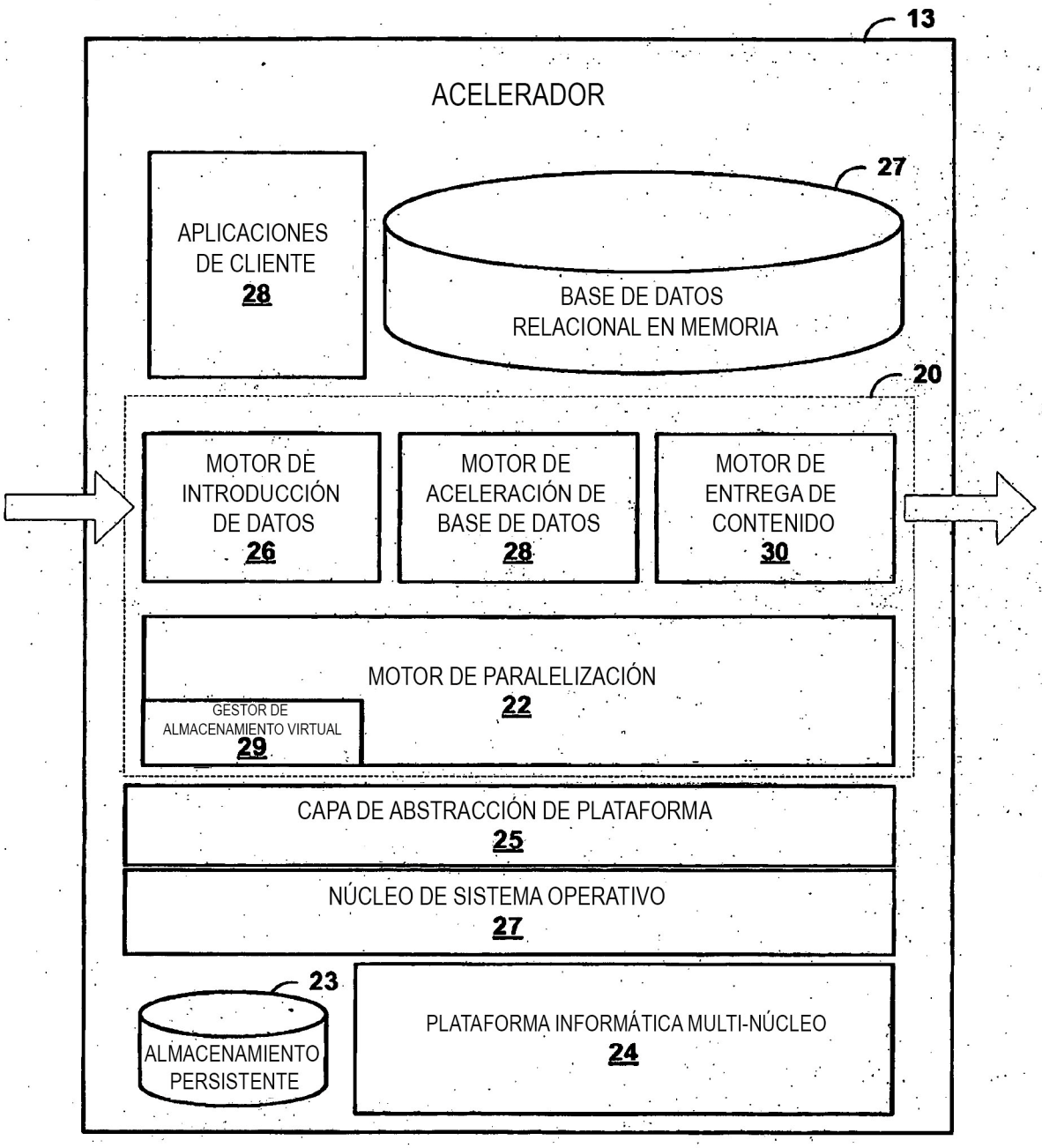
13
ACELERADOR
APLICACIONES DE CLIENTE
28
27
BASE DE DATOS RELACIONAL EN MEMORIA
20
MOTOR DE INTRODUCCIÓN DE DATOS
26
MOTOR DE ACELERACIÓN DE BASE DE DATOS
28
MOTOR DE ENTREGA DE CONTENIDO
30
MOTOR DE PARALELIZACIÓN
22
GESTOR DE ALMACENAMIENTO VIRTUAL
29
CAPA DE ABSTRACCIÓN DE PLATAFORMA
25
NÚCLEO DE SISTEMA OPERATIVO
27
23
ALMACENAMIENTO PERSISTENTE
PLATAFORMA INFORMÁTICA MULTI-NÚCLEO
24

FIG. 2

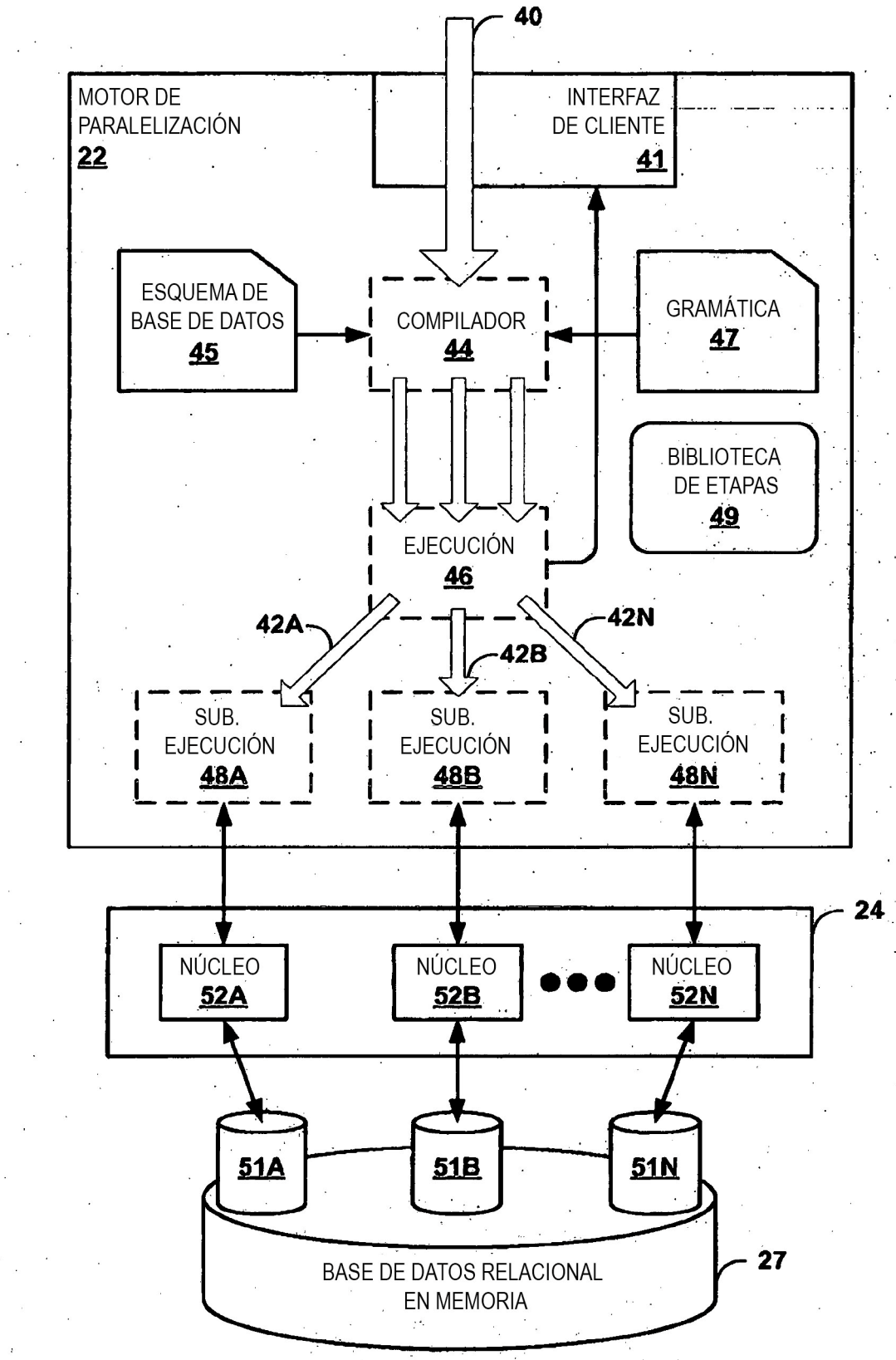
40
MOTOR DE PARALELIZACIÓN
22
INTERFAZ DE CLIENTE
41
ESQUEMA DE BASE DE DATOS
45
COMPILADOR
44
GRAMÁTICA
47
BIBLIOTECA DE ETAPAS
49
EJECUCIÓN
46
42A
42B
42N
SUB. EJECUCIÓN
48A
SUB. EJECUCIÓN
48B
SUB. EJECUCIÓN
48N
24
NÚCLEO
52A
NÚCLEO
52B
NÚCLEO
52N
51A
51B
51N
BASE DE DATOS RELACIONAL EN MEMORIA
27

FIG. 3

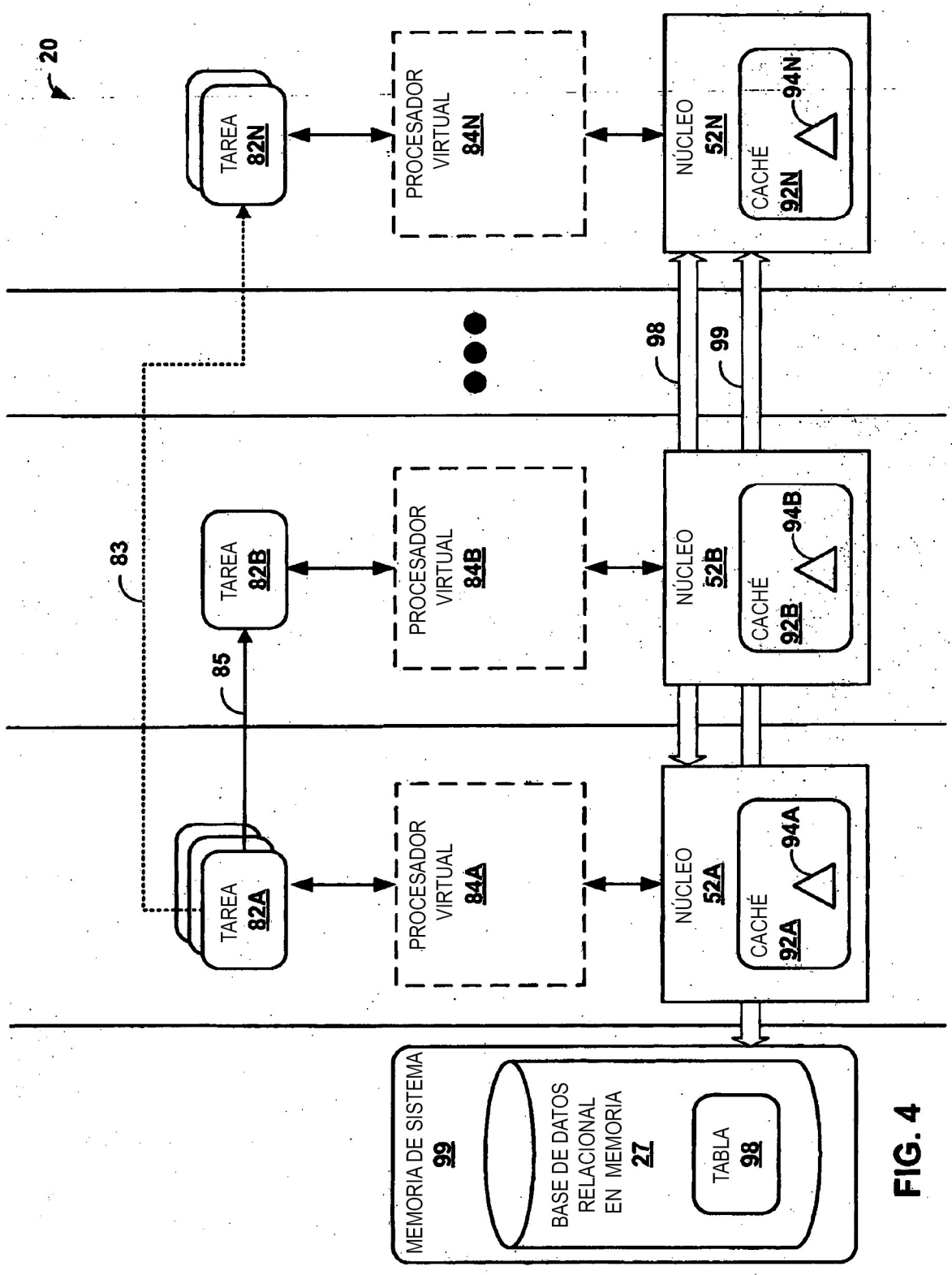
20
TAREA 82N
PROCESADOR VIRTUAL 84N
NÚCLEO 52N
CACHÉ 92N
94N
98
99
83
TAREA 82B
PROCESADOR VIRTUAL 84B
NÚCLEO 52B
CACHÉ 92B
94B
85
TAREA 82A
PROCESADOR VIRTUAL 84A
NÚCLEO 52A
CACHÉ 92A
94A
MEMORIA DE SISTEMA 99
BASE DE DATOS RELACIONAL EN MEMORIA 27
TABLA 98

FIG. 4

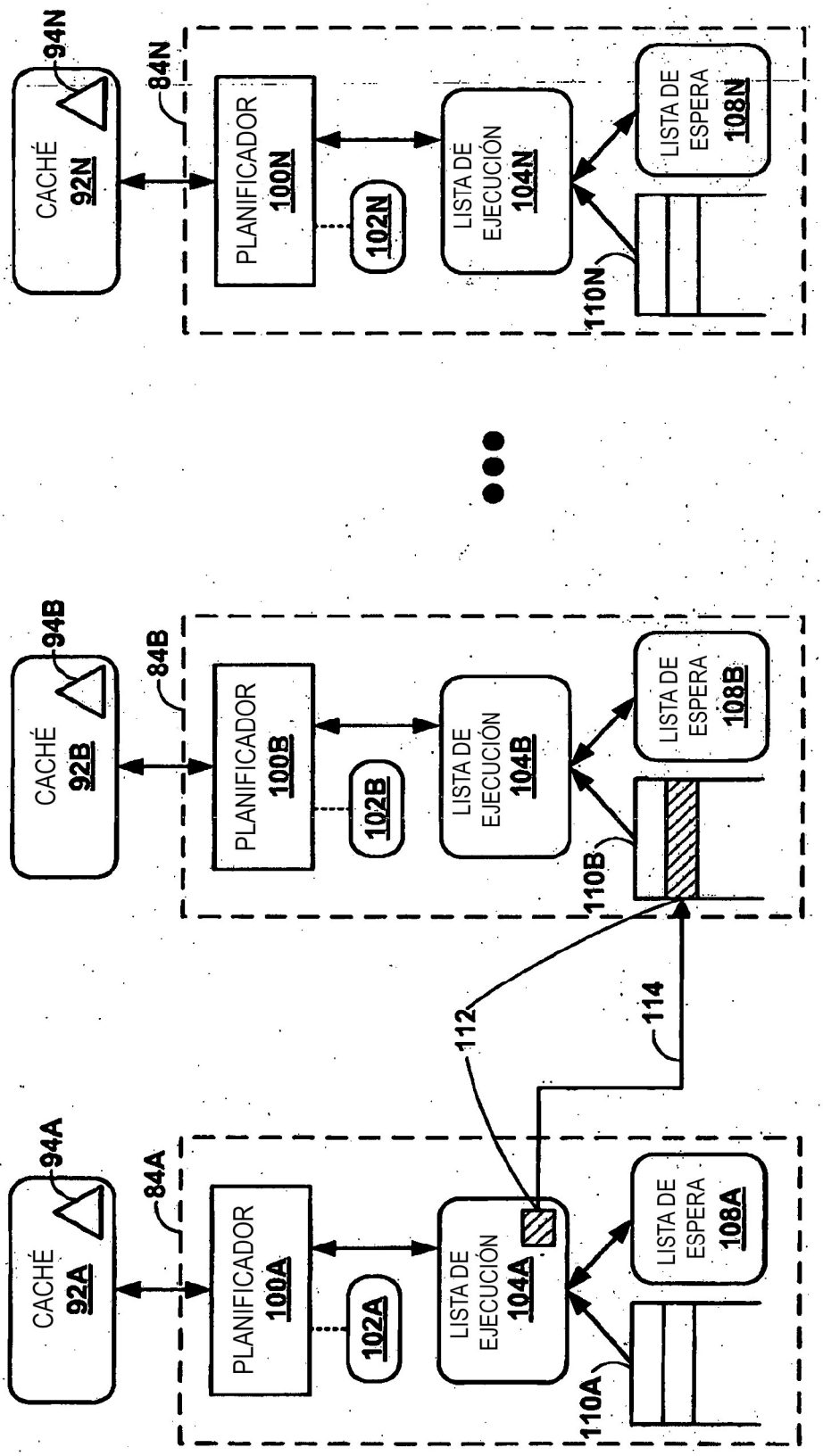
CACHÉ 92A
94A
84A
PLANIFICADOR 100A
102A
LISTA DE EJECUCIÓN 104A
LISTA DE ESPERA 108A
110A
CACHÉ 92B
94B
84B
PLANIFICADOR 100B
102B
LISTA DE EJECUCIÓN 104B
LISTA DE ESPERA 108B
110B
CACHÉ 92N
94N
84N
PLANIFICADOR 100N
102N
LISTA DE EJECUCIÓN 104N
LISTA DE ESPERA 108N
110N
112
114

FIG. 5

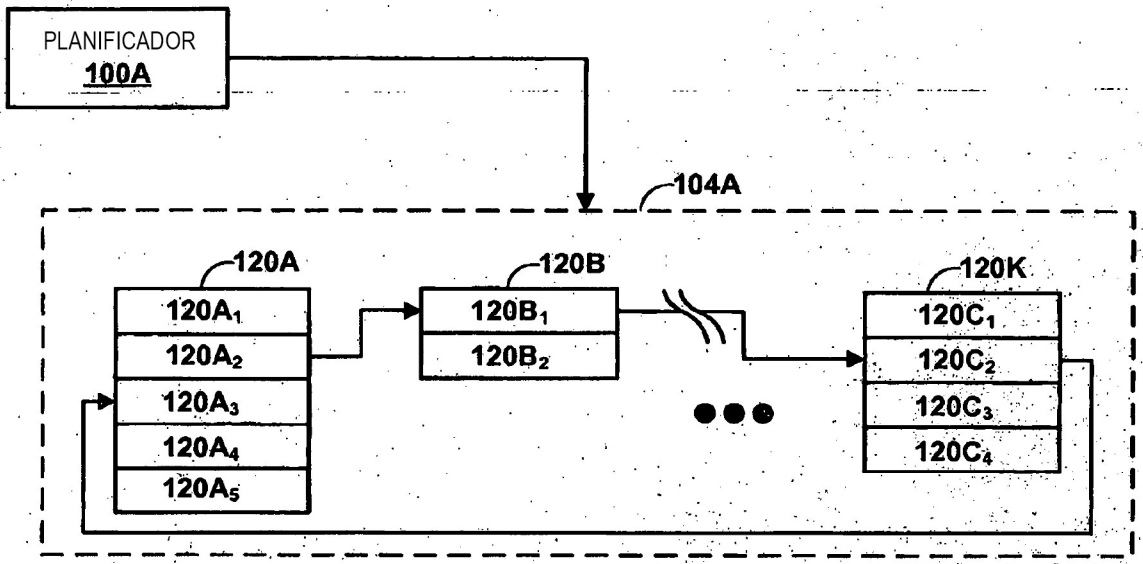
PLANIFICADOR
100A
104A
120A
120A1
120A2
120A3
120A4
120A5
120B
120B1
120B2
120K
120C1
120C2
120C3
120C4

FIG. 6

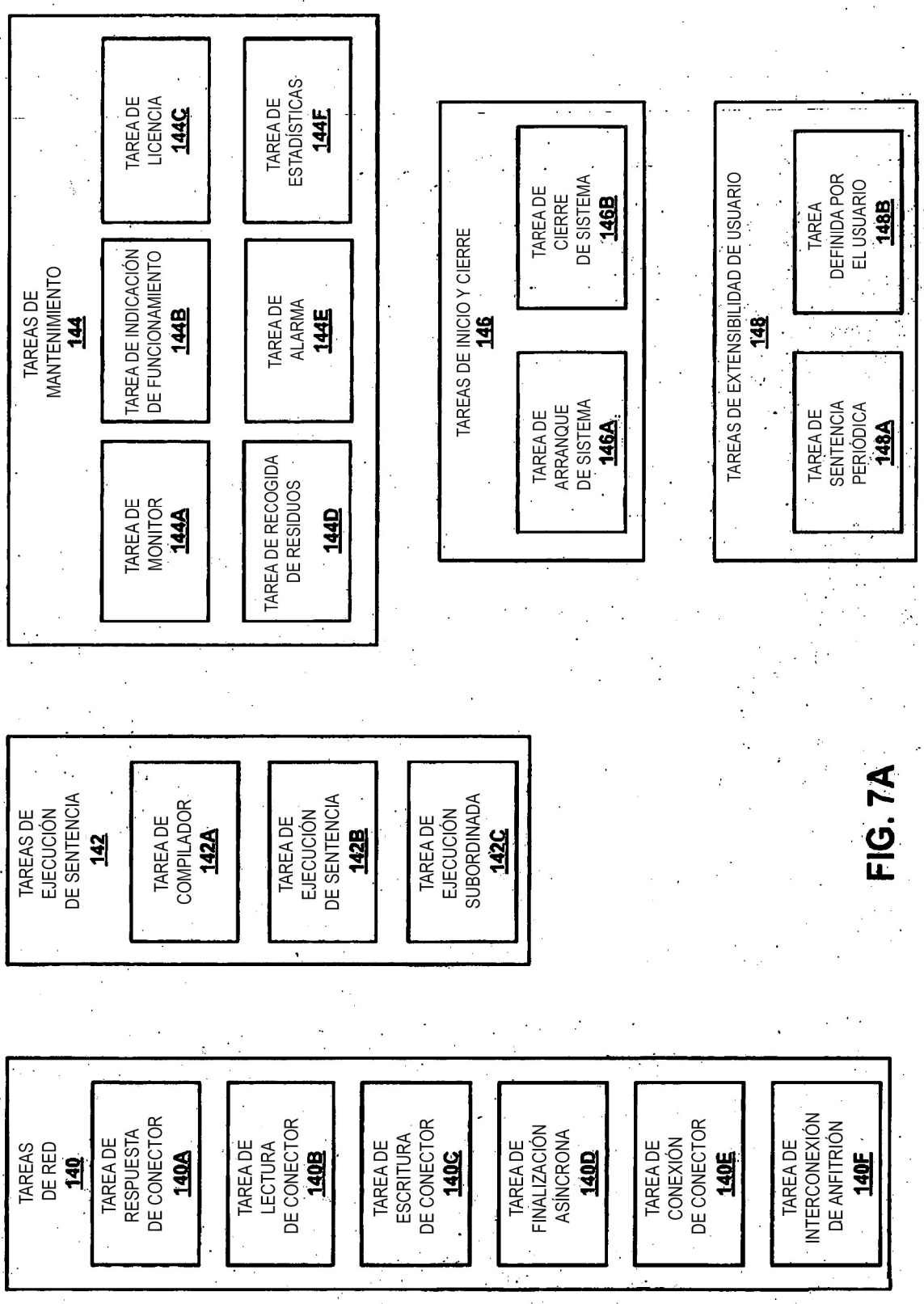
TAREAS DE RED 140
TAREA DE RESPUESTA DE CONECTOR 140A
TAREA DE LECTURA DE CONECTOR 140B
TAREA DE ESCRITURA DE CONECTOR 140C
TAREA DE FINALIZACIÓN ASÍNCRONA 140D
TAREA DE CONEXIÓN DE CONECTOR 140E
TAREA DE INTERCONEXIÓN DE ANFITRIÓN 140F
TAREAS DE EJECUCIÓN DE SENTENCIA 142
TAREA DE COMPILADOR 142A
TAREA DE EJECUCIÓN DE SENTENCIA 142B
TAREA DE EJECUCIÓN SUBORDINADA 142C
TAREAS DE MANTENIMIENTO 144
TAREA DE MONITOR 144A
TAREA DE INDICACIÓN DE FUNCIONAMIENTO 144B
TAREA DE LICENCIA 144C
TAREA DE RECOGIDA DE RESIDUOS 144D
TAREA DE ALARMA 144E
TAREA DE ESTADÍSTICAS 144F
TAREAS DE INICIO Y CIERRE 146
TAREA DE ARRANQUE DE SISTEMA 146A
TAREA DE CIERRE DE SISTEMA 146B
TAREAS DE EXTENSIBILIDAD DE USUARIO 148
TAREA DE SENTENCIA PERIÓDICA 148A
TAREA DEFINIDA POR EL USUARIO 148B

FIG. 7A

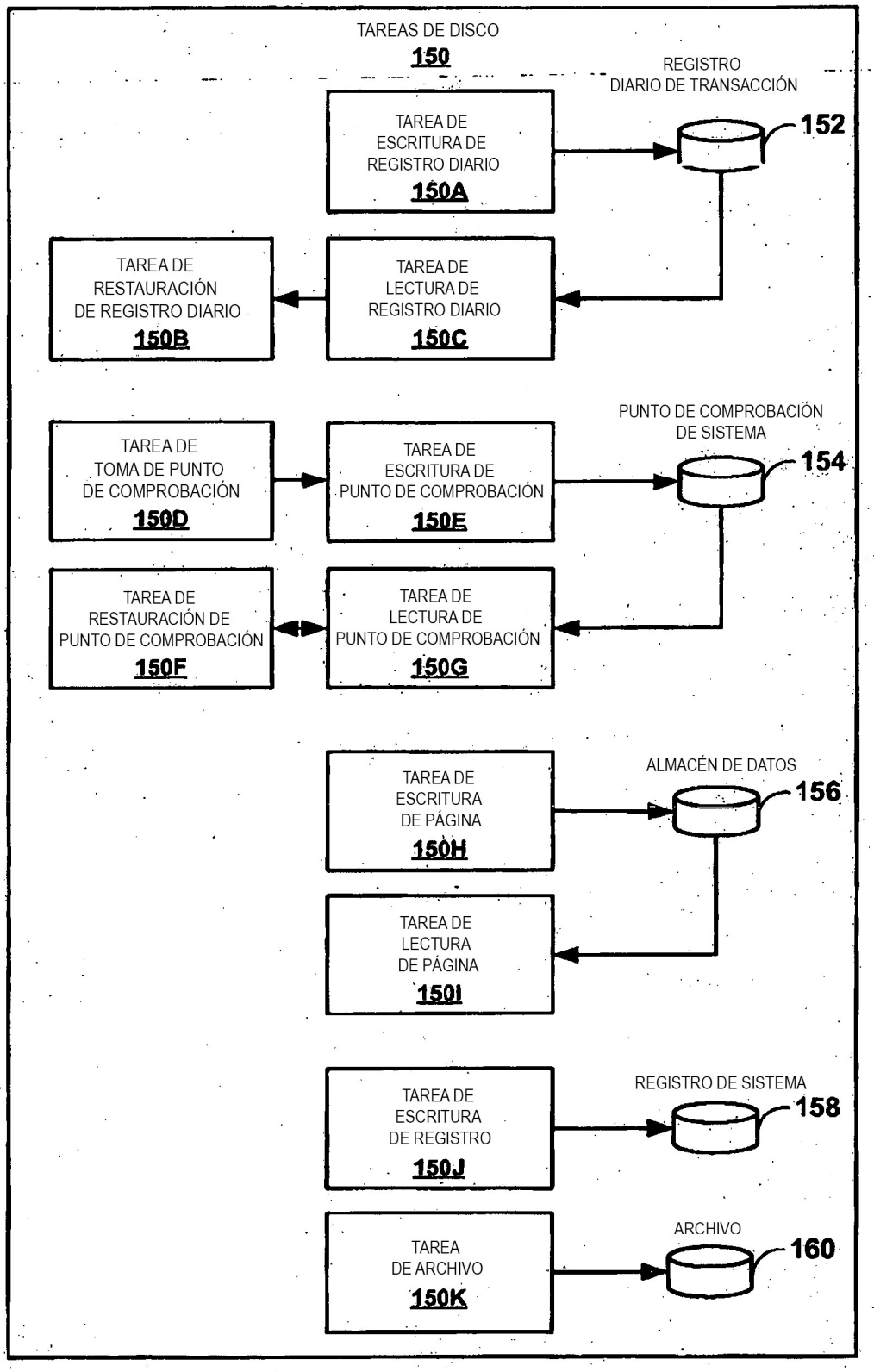
TAREAS DE DISCO
150
REGISTRO DIARIO DE TRANSACCIÓN
152
TAREA DE ESCRITURA DE REGISTRO DIARIO
150A
TAREA DE RESTAURACIÓN DE REGISTRO DIARIO
150B
TAREA DE LECTURA DE REGISTRO DIARIO
150C
PUNTO DE COMPROBACIÓN DE SISTEMA
154
TAREA DE TOMA DE PUNTO DE COMPROBACIÓN
150D
TAREA DE ESCRITURA DE PUNTO DE COMPROBACIÓN
150E
TAREA DE RESTAURACIÓN DE PUNTO DE COMPROBACIÓN
150F
TAREA DE LECTURA DE PUNTO DE COMPROBACIÓN
150G
ALMACÉN DE DATOS
156
TAREA DE ESCRITURA DE PÁGINA
150H
TAREA DE LECTURA DE PÁGINA
150I
REGISTRO DE SISTEMA
158
TAREA DE ESCRITURA DE REGISTRO
150J
ARCHIVO
160
TAREA DE ARCHIVO
150K

FIG. 7B

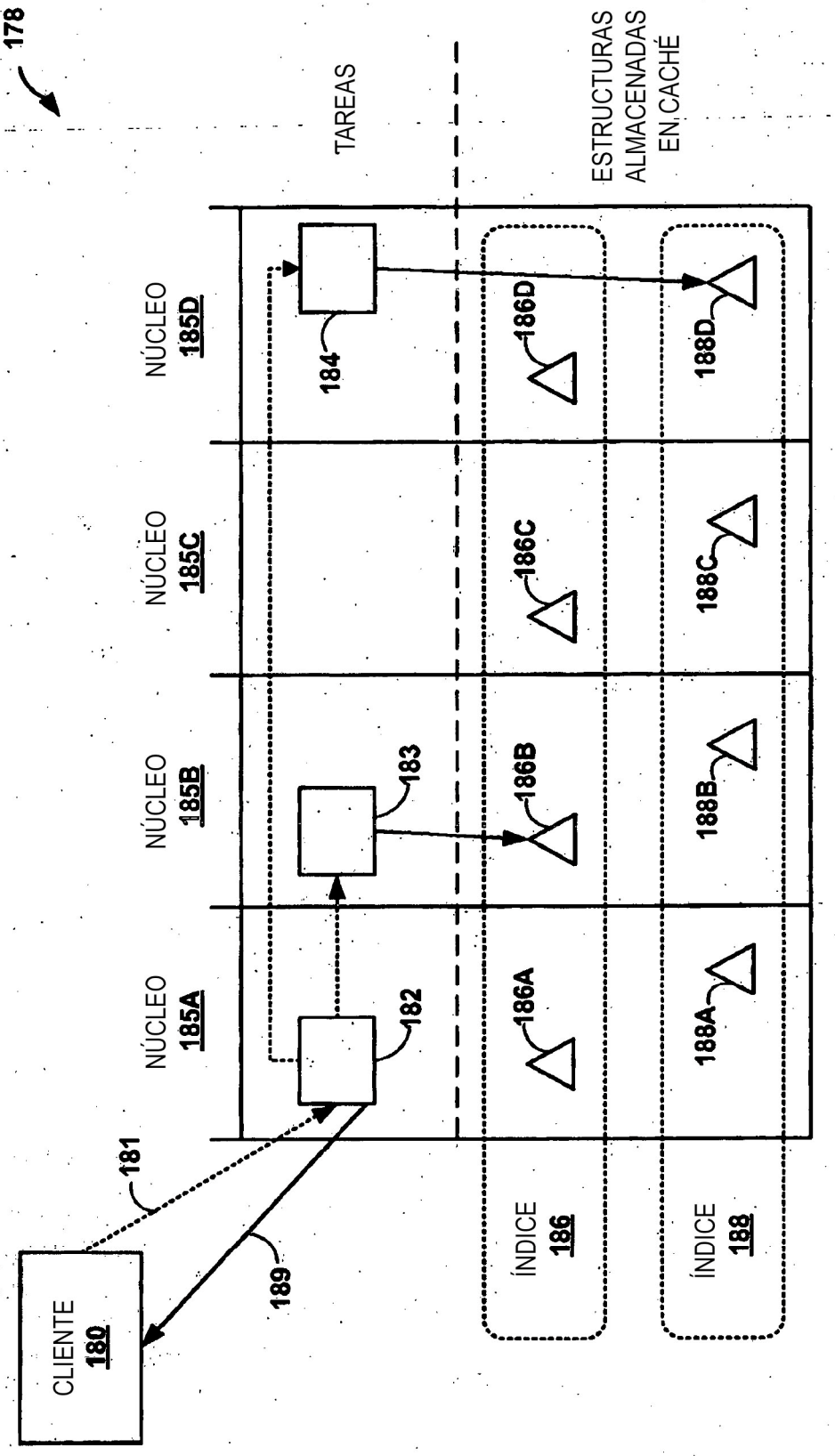
178
TAREAS
ESTRUCTURAS ALMACENADAS EN CACHÉ
NÚCLEO 185D
NÚCLEO 185C
NÚCLEO 185B
NÚCLEO 185A
184
183
182
186D
186C
186B
186A
188D
188C
188B
188A
ÍNDICE 186
ÍNDICE 188
CLIENTE 180
181
189

FIG. 8

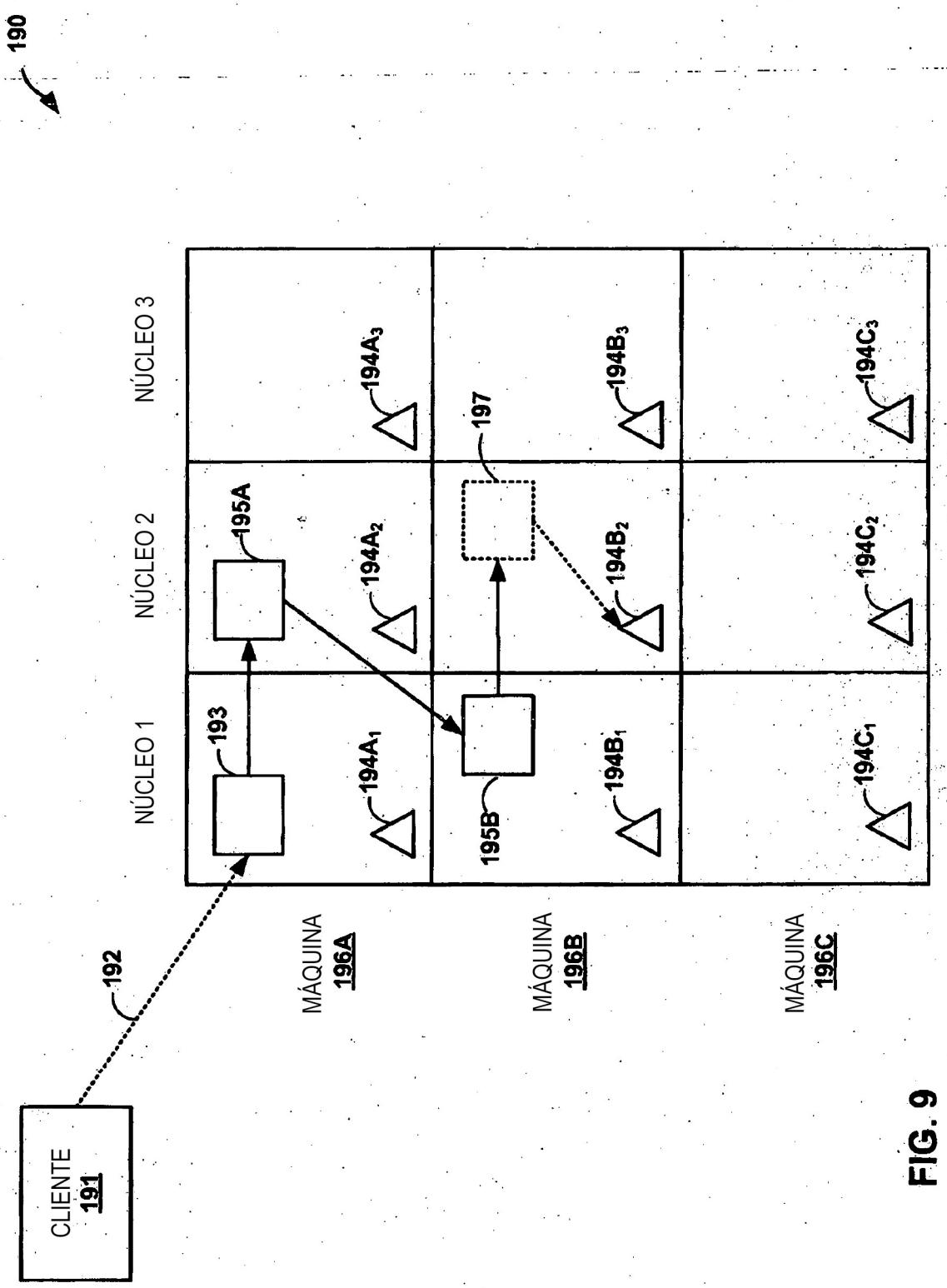
190
NÚCLEO 1
NÚCLEO 2
NÚCLEO 3
193
195A
195B
197
194A1
194A2
194A3
194B1
194B2
194B3
194C1
194C2
194C3
MÁQUINA 196A
MÁQUINA 196B
MÁQUINA 196C
192
CLIENTE 191

FIG. 9

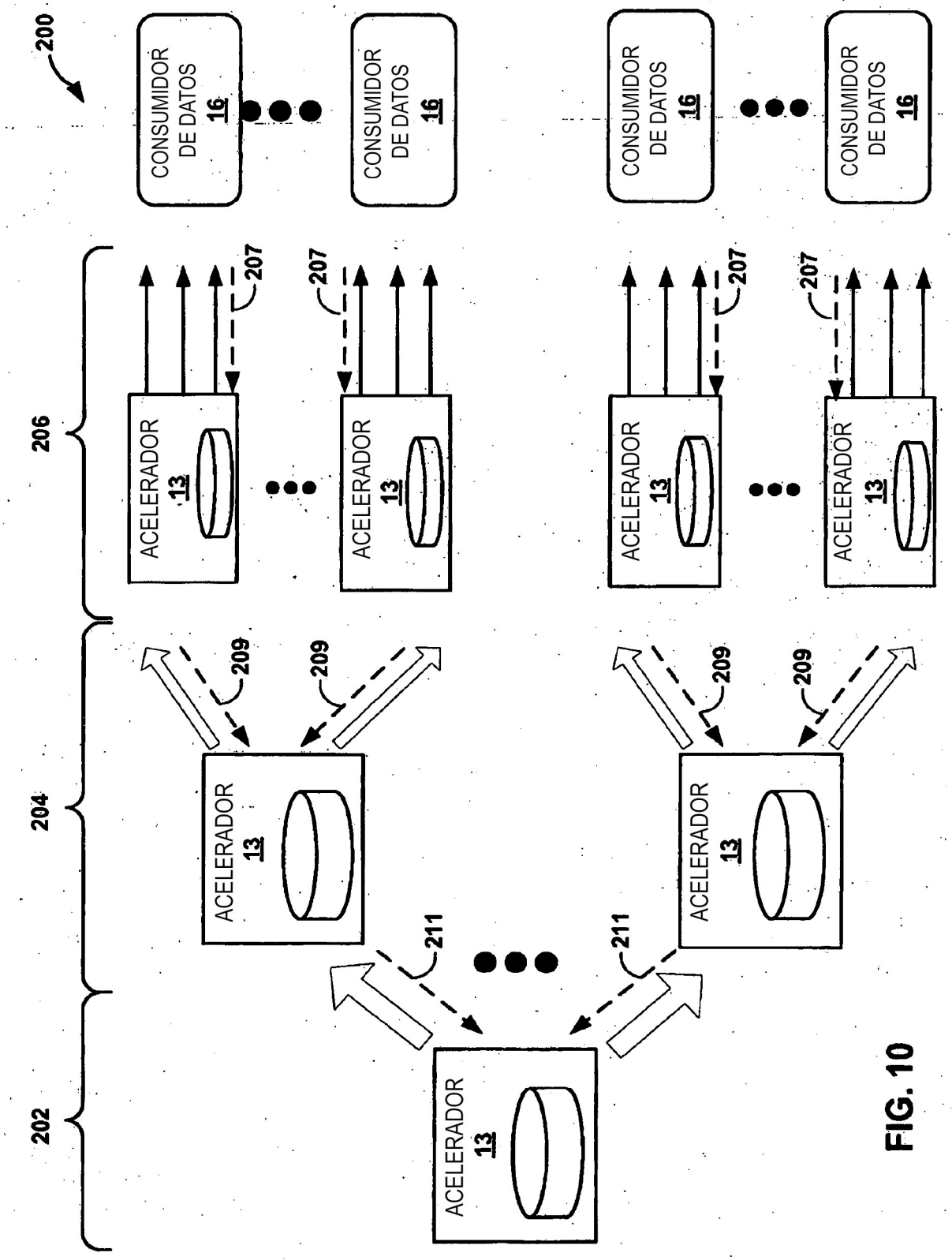
200
CONSUMIDOR DE DATOS 16
CONSUMIDOR DE DATOS 16
CONSUMIDOR DE DATOS 16
CONSUMIDOR DE DATOS 16
206
ACELERADOR 13
ACELERADOR 13
ACELERADOR 13
ACELERADOR 13
207
207
207
207
204
209
209
209
209
211
211
202
ACELERADOR 13

FIG. 10

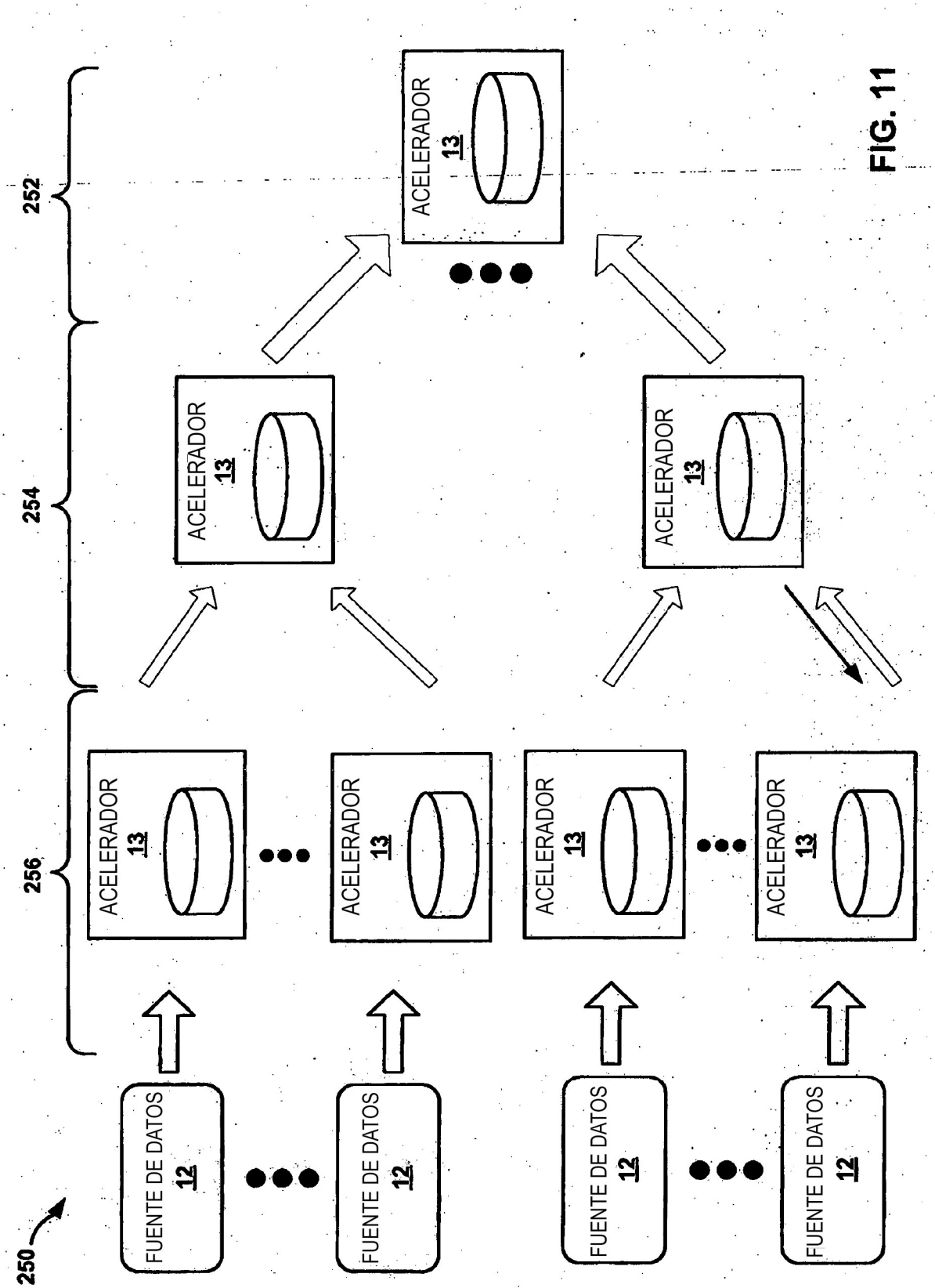

FIG. 11

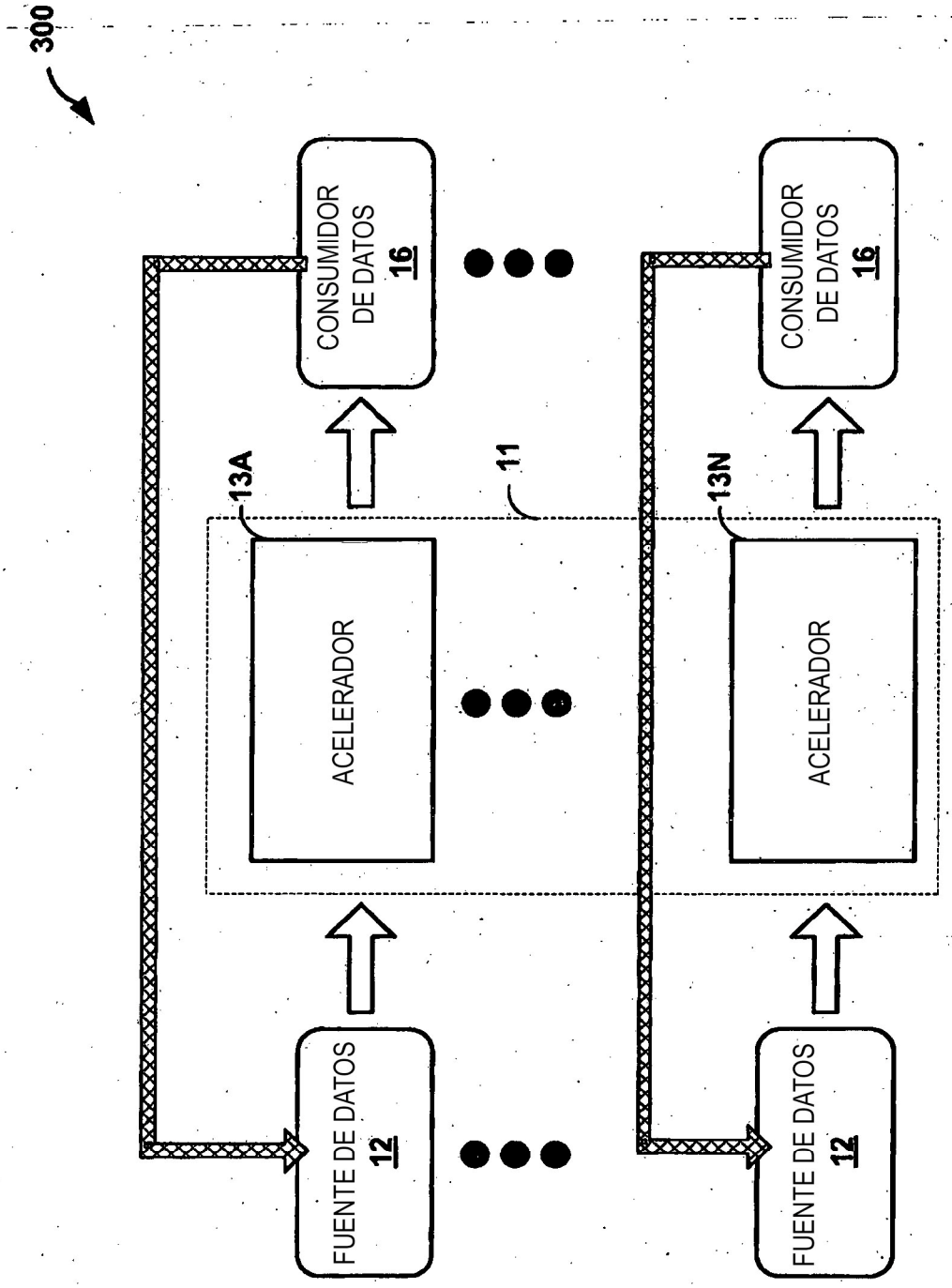
300
CONSUMIDOR DE DATOS 16
CONSUMIDOR DE DATOS 16
13A
11
13N
ACELERADOR
ACELERADOR
FUENTE DE DATOS 12
FUENTE DE DATOS 12

FIG. 12