(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 632 153**

(51) Int. CI.:

G06F 15/16 (2006.01)
G06F 17/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **28.09.2006 PCT/US2006/038117**

(87) Fecha y número de publicación internacional: **26.04.2007 WO07047066**

(96) Fecha de presentación y número de la solicitud europea: **28.09.2006 E 06815819 (5)**

(97) Fecha y número de publicación de la concesión europea: **12.04.2017 EP 1941382**

(54) Título: **Equilibrio de carga**

(30) Prioridad:

20.10.2005 US 254649

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.09.2017

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

STEPHENS, MAONI Z. y
DUSSUD, PATRICK H.

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Equilibrio de carga

Antecedentes

La gestión de memoria en un entorno de ejecución gestionado puede incluir suspender temporalmente hilos gestionados durante un barrido o exploración de una memoria dinámica. Sin embargo, una suspensión prolongada de los hilos gestionados puede pausar la ejecución de la aplicación correspondiente.

El documento WO 00/05652 A desvela un mecanismo para asignación de memoria determinística. De acuerdo con el documento, se pre-asigna espacio de memoria contiguo y se usa para asignaciones de memoria posteriores. Este procedimiento evita la activación de recolección de basura y evita la fragmentación de memoria asignada, mientras que mantiene una asignación de memoria dinámica.

Balakrishnan, S. y col.: "The Impact of Performance Asymmetry in Emerging Multicore Architectures", 4 de junio de 2005, Arquitectura informática, 2005, ISCA '05. Actas. 32do Simposio Internacional en Madison, WI, Estados Unidos, 4-8 de junio de 2005, Piscataway, NJ, Estados Unidos, IEEE, páginas 506-517, se refiere al comportamiento de la ejecución de aplicaciones comerciales en sistemas asimétricos de rendimiento. Presenta el primer estudio que investiga el impacto de asimetría de rendimiento en una amplia variedad de aplicaciones comerciales que usan un prototipo de hardware. La recolección de basura tiene un impacto significativo en la capacidad de procesamiento en un sistema simétrico. El estudio actual usó recolección de basura concurrente de único hilo y un recolector no concurrente de múltiples hilos. Diseños de recolectores de basura futuros deberían tener en cuenta la asimetría de rendimiento subyacente, para garantizar comportamiento de aplicación estable.

Sumario

Es el objeto de la presente invención limitar el efecto de la realización de una recolección de basura a ejecución de código en un entorno de ejecución gestionado.

Este objeto se resuelve mediante el objeto de las reivindicaciones independientes.

Se definen realizaciones preferidas en las reivindicaciones dependientes.

La ejecución de porciones de una aplicación, programa, procedimiento, función u otro conjunto de código en un entorno de ejecución gestionado puede influenciarse mediante, al menos, la magnitud potencial de una implementación de gestión de memoria.

Descripción de los dibujos

Equilibrio de carga en un entorno de ejecución gestionado se describe ahora de acuerdo con las siguientes figuras.

La Figura 1 muestra dispositivos que se comunican en una red, con los dispositivos implementando tecnologías de ejemplo para equilibrio de carga.
La Figura 2 muestra un ejemplo de un entorno de ejecución para la implementación de tecnologías de ejemplo para equilibrio de carga.
La Figura 3 muestra un flujo de datos de ejemplo entre nodos de datos de acuerdo con una implementación de ejemplo de equilibrio de carga.
La Figura 4 muestra un flujo de procesamiento de ejemplo de acuerdo con una implementación de ejemplo de equilibrio de carga.

Descripción detallada

En el presente documento se describe el equilibrio de carga. Más particularmente, la presente descripción se refiere al control de cierto comportamiento de al menos porciones de una aplicación, programa, procedimiento, función u otro conjunto de código a base de datos con respecto a, al menos, la magnitud potencial de una implementación de gestión de memoria en el respectivo entorno de ejecución.

Dentro del contexto de esta descripción detallada y como perteneciente a programación orientada a objetos, un procedimiento puede considerarse como el procesamiento que realiza un objeto. Por lo tanto, por ejemplo, cuando se envía un mensaje a un objeto, se implementa el procedimiento:

"Procedimiento," como se describe en el presente documento, puede referirse a la ejecución real de un módulo o conjunto de código perteneciente a una aplicación, programa, función u otro conjunto de código programable y ejecutable.

"Conjunto," como se describe en el presente documento, puede referirse a una unidad de despliegue de código, que se puede o no versionar.

"Recurso," como se describe en el presente documento, puede incluir tanto recursos físicos como lógicos asociados con un entorno informático dado. Como ejemplos no limitantes, tales recursos pueden variar desde archivos a puertos a estado compartido; es decir, cualquier entidad no ejecutable que puede compartirse mediante más de una entidad ejecutable.

"Hilos," como se describe en el presente documento, puede referirse a trayectorias de ejecución dentro de una aplicación, programa, función u otro conjunto de código programable y ejecutable. Los hilos habilitan que múltiples trayectorias o flujos de ejecución de módulos de instrucciones ejecutables sucedan simultáneamente dentro de la misma aplicación, programa, función u otros conjuntos de código programable y ejecutable; con lo que, dentro de cada flujo, puede procesarse una transacción o mensaje diferentes. Un entono multitarea o multiprocesamiento, en el que pueden ejecutarse procedimientos de múltiples hilos, pueden encontrarse en cualquiera de un entorno de ejecución gestionado o un entorno de ejecución no gestionado.

"Límite de aislamiento," como se describe en el presente documento, puede referirse a una construcción lógica o física que puede servir como una unidad de asilamiento. Los procedimientos son un ejemplo de un límite de aislamiento. Dentro de un entorno de ejecución gestionado, un límite de aislamiento de este tipo puede denominarse como un dominio de aplicación, en el que pueden contenerse múltiples hilos de ejecución. Tal terminología se proporciona únicamente como un ejemplo. Es decir, las implementaciones de ejemplo descritas en el presente documento no se limitan a dominios de aplicación o incluso a entorno de ejecución gestionados como se indicó anteriormente, sino que pueden aplicarse dentro de diversas otras implementaciones de límites de aislamiento en otros entornos de ejecución. Más particularmente, los límites de aislamiento, como se refieren al alcance de distribución de recursos descrito en el presente documento, pueden adicionalmente pertenecer a límites de máquina, límites de procedimiento, hilos y límites de clase o conjunto. Incluso más particularmente, el alcance de distribución de recursos puede pertenecer a exposición pública/privada, conjuntos o clases. Además, distribución de recursos puede tener múltiples ejes o anotaciones que incluyen, por ejemplo, un tipo de recurso así como visibilidad del recurso.

Los límites de aislamiento pueden habilitar que el código se ejecute en el mismo para cargarse desde una fuente específica; un límite de aislamiento puede abortarse independiente de otros tales límites de aislamiento; y el procesamiento dentro de un límite de aislamiento puede aislarse de modo que un fallo que sucede en el mismo no afecta a otros límites de aislamiento dentro del procedimiento. Más particularmente, los límites de aislamiento pueden aislar el consumo de recursos en el mismo hasta el extremo que otros límites de aislamiento o bien no ven ningún cambio en un recurso o, en su lugar, ven los recursos de un modo serializado y atómico.

La Figura 1 muestra entorno 100 de red de ejemplo que tiene nodos de procesamiento que pueden comunicarse entre sí en diversas cantidades y combinaciones de los mismos para implementar uno o más ejemplos de equilibrio de carga. Sin embargo, las implementaciones de equilibrio de carga no se limitan a nodos en un entorno de red. Sin embargo, en el entono de red de ejemplo de la Figura 1, el dispositivo 105 de cliente, dispositivos 110A y 110B de servidor y "otro" dispositivo 115 pueden acoplarse comunicativamente entre sí a través de la red 125; y, además, al menos uno del dispositivo 105 de cliente, dispositivos 110A y 110B de servidor y "otro" dispositivo 115 puede ser capaz de implementar equilibrio 120 de carga, como se describe en el presente documento.

El dispositivo 105 de cliente puede ser al menos uno de una diversidad de dispositivos informáticos convencionales, incluyendo un ordenador personal (PC) de sobremesa, estación de trabajo, ordenador central, aparato de Internet y decodificadores de salón. Además, el dispositivo 105 de cliente puede ser al menos uno de cualquier dispositivo que es capaz de asociarse con red 125 mediante un enlace cableado y/o inalámbrico, incluyendo un asistente digital personal (PDA), ordenador portátil, teléfono celular, etc. Aún más, el dispositivo 105 de cliente puede representar los dispositivos de cliente descritos anteriormente en diversas cantidades y/o combinaciones de los mismos. "Otro" dispositivo 115 también puede incorporarse mediante cualquiera de los ejemplos anteriores de dispositivo 105 de cliente.

Los dispositivos 110A y 110B de servidor pueden proporcionar cualquiera de una diversidad de datos y/o funcionalidad al dispositivo 105 de cliente u "otro" dispositivo 115 de acuerdo con al menos una implementación de equilibrio 120 de carga. Los datos pueden estar disponibles públicamente o como alternativa restringidos, por ejemplo, restringidos a únicamente ciertos usuarios o únicamente si se paga una tarifa de suscripción o de licencia apropiada. Cualquiera de los dispositivos 110A y 110B de servidor puede ser un servidor de red, un servidor de aplicación o un servidor laminar, en diversas cantidades y combinaciones de los mismos. Típicamente, los dispositivos 110A y 110B de servidor sirven como fuentes de contenido y el dispositivo 105 de cliente recibe tal contenido o bien través de la red 125 o de una manera fuera de línea. Sin embargo, de acuerdo con las implementaciones de ejemplo descritas en el presente documento, el dispositivo 105 de cliente y dispositivos 110A y 110B de servidor pueden estar enviado de forma intercambiable nodos o recibiendo nodos en el entorno 100 de red. Además, de acuerdo con al menos un ejemplo de equilibrio 120 de carga, los dispositivos 110A y 110B de servidor pueden implementarse como uno de muchos dispositivos de servidor en la red 125. Tal configuración puede considerarse, informalmente, como una "torre de servidores." En una torre de servidores de este tipo, los dos o más servidores interconectados pueden compartir en la ejecución de al menos porciones de una misma aplicación, programa, función u otro conjunto de código programable y ejecutable. Aún más, "otro" dispositivo 115 también puede incorporarse mediante cualquiera de los anteriores ejemplos de dispositivos 110A y 110B de servidor.

"Otro" dispositivo 115 puede ser cualquier dispositivo adicional que es capaz de implementar equilibrio 120 de carga de acuerdo con uno o más de los ejemplos descritos en el presente documento. Es decir, "otro" dispositivo 115 puede ser un dispositivo informático habilitado para software o de procesamiento que es capaz de implementar equilibrio 120 de carga para al menos una porción de una aplicación, programa, función u otro conjunto de código programable y ejecutable en al menos un entorno de ejecución gestionado. Más particularmente, "otro" dispositivo 115 puede servir como un equilibrador de carga. Como un equilibrador de carga, "otro" dispositivo 115 puede implementarse como un dispositivo de hardware que está físicamente separado de cualquiera de dispositivo 105 de cliente o dispositivos 110A y 110B de servidor; como alternativa, "otro" dispositivo 115 puede implementarse como firmware o software en al menos uno del dispositivo 105 de cliente o dispositivos 110A y 110B de servidor. Por lo tanto, "otro" dispositivo 115 puede ser un dispositivo informático, de procesamiento o servidor que tiene al menos uno de un sistema operativo, un intérprete, convertidor, compilador o entorno de ejecución de tiempo de ejecución implementado en el mismo. No se pretende que estos ejemplos sean limitantes en ninguna manera y por lo tanto no debería interpretarse de esa manera.

La red 125 puede representar cualquiera de una diversidad de topologías y tipos de red convencionales, que pueden incluir redes cableadas y/o inalámbricas. La red 125 puede utilizar adicionalmente cualquiera de una diversidad de protocolos de red convencionales, incluyendo protocolos públicos y/o propietarios. La red 125 puede incluir, por ejemplo, la Internet así como al menos porciones de una o más redes de área local (también denominadas, individualmente, como una "LAN"), tal como un sistema 802.11; una red de área personal (es decir, PAN), tal como Bluetooth.

La arquitectura informática en al menos uno de los dispositivos 105, 110A, 110B y 115 habitualmente ha definido plataformas informáticas en términos de hardware y software. Software para dispositivos informáticos se categorizan en grupos, a base de función, que incluye: una capa de abstracción de hardware (como alternativa denominada como una "HAL"), un sistema operativo (como alternativa denominado como "OS") y aplicaciones.

Un entorno de ejecución de tiempo de ejecución puede referirse a un espacio aislado, que puede estar entre el OS y una aplicación, en el que la aplicación puede ejecutar tareas específicas en al menos uno de dispositivo de procesamiento 105, uno o más de dispositivos 110A y 110B de servidor u otro dispositivo 115. Más particularmente, un entorno de ejecución de tiempo de ejecución se concibe para mejorar la fiabilidad de la ejecución de aplicaciones en una variedad en aumento de dispositivos de procesamiento que incluyen servidores, ordenadores de sobremesa, ordenadores portátiles, dispositivos de procesamiento móviles, decodificadores de salón y consolas de juegos proporcionando una capa de abstracción y servicios para una aplicación en ejecución en tales dispositivos de procesamiento y adicionalmente proporcionando a la aplicación con capacidades que incluyen la gestión de memoria y configuración de la misma.

Un entorno de ejecución de tiempo de ejecución puede servir como al menos una de una plataforma de programación de aplicación y de ejecución de aplicación.

Como una plataforma de programación de aplicación, un entorno de ejecución de tiempo de ejecución puede compilar aplicaciones objetivo, que pueden escribirse en uno de múltiples lenguajes informáticos, en un lenguaje intermedio (en lo sucesivo "IL"). El IL es habitualmente independiente de la plataforma y la unidad de procesamiento central (en lo sucesivo "CPU") ejecuta el IL. De hecho, el IL es un lenguaje de nivel superior que muchos lenguajes de máquina de CPU.

Como una plataforma de ejecución de aplicación, un entorno de ejecución de tiempo de ejecución puede interpretar IL compilado en instrucciones de máquina nativas. Un entorno de ejecución de tiempo de ejecución puede utilizar o bien un intérprete o un compilador "justo a tiempo" (en lo sucesivo "JIT") para ejecutar tales instrucciones. Como alternativa, un entorno de ejecución de tiempo de ejecución puede ejecutar, leer, interpretar o de otra manera analizar código de lenguaje intermedio (es decir, "IL") que se distribuye al entorno de ejecución de tiempo de ejecución en formato IL en vez de en un formato de ejecución de plataforma nativo y ya está compilado en uno cualquiera de conjuntos, procedimientos o tipos. Una fuente de tal IL puede disponerse en cualquiera de un entorno de ejecución no gestionado o una implementación separada del entorno de ejecución de tiempo de ejecución en un mismo o separado uno de dispositivos 105, 110, y 115. La fuente puede desplegar el IL en, o antes de, el momento de instalación para la aplicación, programa, procedimiento, función u otro conjunto de código programable y ejecutable al que corresponda el IL.

Independientemente, las instrucciones de máquina nativas pueden entonces ejecutarse directamente mediante la CPU. Ya que el IL es independiente de la CPU, el IL puede ejecutarse en una plataforma de CPU siempre que el OS en ejecución en esa plataforma de CPU aloje un entorno de ejecución de tiempo de ejecución apropiado. Ejemplos de entornos de tiempo de ejecución, a los que puede pertenecer el equilibrio 120 de carga, incluyen: entorno de tiempo de ejecución de Visual Basic; entorno de tiempo de ejecución de Máquina Virtual Java® que se usa para ejecutar, por ejemplo, rutinas de Java®; o Tiempo de Ejecución de Lenguaje Común (CLR) para compilar, por ejemplo, aplicaciones .NET™ de Microsoft en lenguaje de máquina antes de la ejecución de una rutina de llamada. Sin embargo, tal listado proporciona únicamente ejemplos. Las implementaciones de ejemplo no se limitan a solo estos entornos de ejecución gestionados. Además, las implementaciones de ejemplo no se limitan solo a entorno de ejecución gestionados, para uno o más ejemplos pueden implementarse dentro de entornos de pruebas y/o entorno

de ejecución no gestionados.

Una aplicación compilada en IL puede denominarse como "código gestionado," y por lo tanto un entorno de ejecución de tiempo de ejecución puede como alternativa denominarse como un "entorno de ejecución gestionado." Porciones de código gestionado puede denominarse como una "imagen gestionada." Código que no utiliza un entorno de ejecución de tiempo de ejecución para ejecutar puede denominarse como aplicaciones de código nativas.

La Figura 2 muestra un ejemplo de entorno 200 de ejecución de tiempo de ejecución en el que pueden implementarse ejemplos de equilibrio 120 de carga (véase la Figura 1).

De acuerdo con al menos una implementación de ejemplo, el entorno 200 de ejecución de tiempo de ejecución puede facilitar ejecución de código gestionado para una plataforma de dispositivo informático. Código gestionado puede considerarse como parte de un conjunto básico de tecnologías de desarrollo de aplicaciones y adicionalmente puede considerarse como código que se compila para ejecución en entorno 200 de ejecución de tiempo de ejecución para proporcionar un servicio correspondiente a la plataforma de dispositivo informático. Además, el entorno 200 de ejecución de tiempo de ejecución puede traducir código gestionado en un nivel interpretativo en instrucciones que puede intermediarse y a continuación ejecutarse mediante un procesador. Un contexto para el entorno 200 de ejecución de tiempo de ejecución también puede proporcionar librerías de clases, que pueden considerarse como bloques de construcción de software para aplicaciones gestionadas.

De acuerdo con una implementación de ejemplo adicional, el entorno 200 de ejecución de tiempo de ejecución puede proporcionar al menos funcionalidad parcial que de otra manera puede esperarse de un núcleo, que puede o no puede faltar de una plataforma de dispositivo informático dependiendo de restricciones de recurso para uno cualquiera particular de dispositivos 105, 110A, 110B y 115. Por lo tanto, al menos un ejemplo de entorno 200 de ejecución de tiempo de ejecución puede implementar lo siguiente: gestión de rutina de entrada/salida (en lo sucesivo "I/O"), gestión de memoria, administración y gestión de rutina de servicio. Por lo tanto, el entorno 200 de ejecución de tiempo de ejecución puede incluir componente 205 de I/O, al menos un gestor 210 de memoria, administrador 215 y componente 220 de ejecución. Estos componentes, que se describirán en más detalle a continuación, se proporcionan únicamente como ejemplos; es decir, no se pretende que los ejemplos sean limitantes a ninguna implementación particular y no debería hacerse tal inferencia. Además, los componentes pueden implementarse en ejemplos de entorno 200 de ejecución de tiempo de ejecución en diversas combinaciones y configuraciones de los mismos.

El componente 205 de I/O de entorno 200 de ejecución de tiempo de ejecución puede proporcionar acceso asíncrono a fuentes de datos (es decir, procesador y periféricos) asociados con la plataforma de dispositivo informático. Ejemplos de tales fuentes de datos pueden ser uno o más de dispositivos 105, 110A, 110B y 115, descritos anteriormente con respecto a la Figura 1. Más particularmente, el componente 205 de I/O puede proporcionar entorno 200 de ejecución de tiempo de ejecución con capacidad de procesamiento de sistema robusto y rendimiento de línea de flujo adicional de código desde el que se origina una solicitud de I/O.

El gestor 210 de memoria puede referirse a un módulo dentro de o asociado con el entorno 200 de ejecución de tiempo de ejecución que se considera como un "recolector de basura." La recolección de basura (como alternativa denominada en lo sucesivo como "GC") puede considerarse como una prestación robusta de entornos de ejecución de código gestionado por los que se libera un objeto (es decir, se desasigna) si un objeto ya no se usa por ninguna aplicación, tras un barrido o exploración de una memoria dinámica. En al menos un ejemplo de gestor 210 de memoria, un barrido de memoria dinámica libre puede implementarse como una búsqueda lineal. Tal implementación puede ser muy adecuada para un ejemplo de una plataforma de dispositivo informático para la que se limita el tamaño de memoria y para la que puede percibirse un retardo en la finalización de un barrido mediante un usuario de un dispositivo correspondiente.

Un ejemplo de gestor 210 de memoria puede implementar funcionalidad de "GC concurrente" para permitir que hilos gestionados continúen ejecutándose durante un barrido o exploración de una memoria dinámica. Es decir, durante la desasignación, GC concurrente puede permitir que asignación continúe en paralelo. La funcionalidad de GC concurrente (como alternativa denominada en el presente documento como "GC concurrente") puede ser adecuada para una aplicación interactiva o basada en GUI (interfaz de usuario gráfica).

Sin embargo, aunque equilibrio 120 de carga puede ser relevante para funcionalidad de GC concurrente, las implementaciones de ejemplo presentes de equilibrio 120 de carga se describen en el contexto de funcionalidad GC no concurrente, por lo que uno o más hilos gestionados pueden suspenderse durante un barrido o exploración de una memoria dinámica.

Funciones adicionales implementadas por gestor 210 de memoria puede incluir: gestionar uno o más bloques contiguos de almacenamiento de RAM volátil finita (es decir, memoria dinámica) o un conjunto de bloques contiguos de memoria entre las funciones en ejecución en la plataforma de dispositivo informático; asignar memoria a al menos una aplicación en ejecución en la plataforma de dispositivo informático; liberar al menos porciones de memoria a petición de por al menos una de las aplicaciones; y evitar que cualquiera de las aplicaciones acceda intrusivamente a espacio de memoria que ha sido asignado a cualquiera de las otras aplicaciones.

El administrador 215 puede referirse a un módulo dentro de o asociado con el entorno 220 de ejecución de tiempo de ejecución que sirve para recibir al menos una porción de una aplicación, programa, procedimiento, función u otro conjunto de código programable y ejecutable para ejecución en entorno 200 de ejecución de tiempo de ejecución. Además, de acuerdo con al menos una implementación de ejemplo de equilibrio 120 de carga, el administrador 215 puede coordinar con el gestor 210 de memoria para controlar el comportamiento de la aplicación, programa, procedimiento, función u otro conjunto de código programable y ejecutable dentro del entorno 220 de ejecución de tiempo de ejecución, en tiempo de compilación, tiempo de ejecución inicial o en cualquier tiempo posteriormente durante ejecución de una aplicación.

El componente 220 de ejecución puede referirse a un módulo dentro de o asociado con el entorno 200 de ejecución de tiempo de ejecución que sirve para habilitar la ejecución de código gestionado por la plataforma de dispositivo informático. El componente 220 de ejecución puede considerarse como el entorno en el que se implementa la ejecución del código de la aplicación y en el que puede proporcionarse servicios de tiempo de ejecución (por ejemplo, acceso de dispositivo y gestión de memoria).

La Figura 3 muestra flujo 300 de datos de ejemplo de acuerdo con una implementación de ejemplo de equilibrio 120 de carga (véase la Figura 1). El flujo 300 de datos de ejemplo se describe ahora con referencia a prestaciones de las Figuras 1 y 2, aunque tales implementaciones se proporcionan únicamente como ejemplos y no se pretende que se interpreten de ninguna manera limitante.

Los procedimientos 305A, 305B y 305C pueden considerarse como construcciones lógicas o físicas, de acuerdo con al menos la descripción de "límites de aislamiento" proporcionada anteriormente. Por lo tanto, los procedimientos 305A, 305B y 305C pueden pertenecer a límites de máquina, límites de procedimiento, hilos, dominios de aplicación y límites de clase o conjunto, de acuerdo con diversas implementaciones de equilibrio 120 de carga. Además, las implementaciones de equilibrio 120 de carga no están de ninguna forma limitadas a tales tres construcciones (es decir, 305A, 305B y 305C), que se ilustran y explican en el presente documento únicamente para fines descriptivos.

Como un ejemplo, los procedimientos 305A, 305B y 305C puede pertenecer a múltiples servidores (es decir, límites de máquina). Implementados como hardware, los servidores 305A, 305B y 305C pueden ser dispositivos de hardware separados, en los que al menos un entorno 200 de ejecución de tiempo de ejecución se ejecuta como una plataforma de ejecución de aplicación. Implementados como servidores de software, los procedimientos 305A, 305B y 305C pueden ser componentes modularizados de una aplicación común, programa, procedimiento, función u otro conjunto de código programable y ejecutable que se ejecuta mediante una implementación del entorno 200 de ejecución de tiempo de ejecución.

La gestión 310 de memoria datos, que se origina de al menos uno de los procedimientos 305A, 305B y 305C, puede incluir datos haciendo referencia a un barrido o exploración inminentes de una memoria dinámica (es decir, GC) para liberar (es decir, desasignar) uno o más objetos que ya no se usan por una aplicación, programa, procedimiento, función u otro conjunto de código que se ejecuta mediante una implementación del entorno 200 de ejecución de tiempo de ejecución.

Mediante el ejemplo en el que se implementan los procedimientos 305A, 305B y 305C como dispositivos de hardware separados, la gestión 310 de memoria datos puede originarse de una implementación del entorno 200 de ejecución de tiempo de ejecución en ejecución en uno o más de tales dispositivos. Mediante el ejemplo en el que los procedimientos 305A, 305B y 305C se implementan como componentes modularizados de una aplicación común, programa, procedimiento, función u otro conjunto de código programable y ejecutable, los datos 310 de gestión de memoria pueden originarse de una implementación del entorno 200 de ejecución de tiempo de ejecución en la que los componentes modularizados se están ejecutando. Más particularmente, mediante los ejemplos anteriores, la gestión 310 de memoria datos puede originarse mediante el gestor 210 de memoria o módulo 215 de administrador, ya sea singularmente o combinado.

La gestión 310 de memoria datos puede implementar transacciones usando una o más interfaces de programa de aplicación (API) que son compatibles con API de diversas arquitecturas de programa. Más particularmente, las API correspondientes a gestión 310 de memoria datos pueden ser capaces de iniciar equilibrio de carga entre los procedimientos 305A, 305B y 305C (es decir, a través de límites de máquina, límites de procedimiento, hilos, dominios de aplicación y límites de clase o conjunto) de acuerdo con diversas implementaciones de equilibrio 120 de carga. Como se describe en el presente documento, una API puede considerarse como una o más rutinas usadas por una aplicación, programa, procedimiento, función u otro conjunto de código programable y ejecutable para dirigir el rendimiento de procedimientos mediante el entorno 200 de ejecución de tiempo de ejecución o incluso un sistema operativo.

La gestión 310 de memoria datos puede incluir una o más API implementadas por el gestor 210 de memoria o módulo 215 de administrador, ya sea singularmente o combinado. La nomenclatura de tales API listadas y descritas a continuación se proporciona para fines descriptivos y se proporcionan como ejemplos no limitantes.

MaxGenerationNotifier puede referirse a una clase de cuyas instanciaciones puede pretenderse que impulsen al equilibrador 315 de carga para redistribuir al menos una porción de la misma u otra aplicación, programa,

procedimiento, función u otro conjunto de código que se ejecuta mediante una implementación particular del entorno 200 de ejecución de tiempo de ejecución. Más particularmente, MaxGenerationNotifier puede instanciarse cuando el gestor 210 de memoria o administrador 215, ya sea singularmente o combinado, detecta que una GC inminente en el entorno 200 de ejecución de tiempo de ejecución puede ejecutarse tanto como para provocar una pausa en la ejecución de la misma u otra aplicación, programa, procedimiento, función u otro conjunto de código en el entorno 200 de ejecución de tiempo de ejecución. Tal detección puede basarse en la asignación de la cantidad umbral de memoria dinámica durante la ejecución actual de la aplicación, programa, procedimiento, función u otro conjunto de código. Efectivamente, a continuación, una instanciación de MaxGenerationNotifier puede proporcionar equilibrador de carga con una notificación proactiva del estado de entorno 200 de ejecución de tiempo de ejecución.

Más particularmente, MaxGenerationNotifier puede instanciarse cuando el gestor 210 de memoria o administrador 215, ya sea singularmente o combinado, detecta que una GC inminente puede potencialmente exceder una magnitud lógica o física de umbral. Por ejemplo, MaxGenerationNotifier puede instanciarse cuando el gestor 210 de memoria o administrador 215, ya sea singularmente o combinado, detecta que la GC inminente es una implementación de una GC de última generación, y por lo tanto es probable que barra o explore la mayor cantidad de memoria dinámica (por ejemplo, 2 GB) permitida en las capacidades de procesamiento disponibles en la actualidad. Como resultado, ejecución de una aplicación, programa, procedimiento, función u otro conjunto de código puede pausar antes de la finalización de la GC de última generación ya que hilos gestionados pueden suspenderse durante un barrido o exploración de una memoria dinámica. Por lo tanto, antes de que empiece una GC de tal magnitud, MaxGenerationNotifier puede instanciarse para notificar al equilibrador 315 de carga de al menos una necesidad para redistribuir al menos una porción de la ejecución de la aplicación, programa, procedimiento, función u otro conjunto de código.

Por supuesto, la memoria dinámica de 2GB se describe anteriormente únicamente como un ejemplo de una GC de última generación. La magnitud de una GC inminente que puede actuar como un catalizador para instanciación de MaxGenerationNotifier puede variar a base de un sinfín de factores y puede aumentar a medida que las capacidades de procesamiento evolucionan. Tales factores pueden referirse a capacidades de procesamiento del dispositivo de procesamiento así como tiempo anticipado para que finalice la GC. Además, la preferencia del programador que escriba la aplicación ejecutable, programa, procedimiento, función u otro conjunto de código, como se indica al administrador 215, también puede iniciar una instanciación de MaxGenerationNotifier, cuando el programador se motiva mediante al menos uno de eficiencia y cuestiones de seguridad.

Aún más, al menos una implementación alternativa de MaxGenerationNotifier puede incluir una o más instrucciones específicas para redirigir ejecución para al menos una porción de la correspondiente aplicación, programa, procedimiento, función u otro conjunto de código que se ejecuta mediante el entorno 200 de ejecución de tiempo de ejecución.

MaxGenerationCompleteNotifier puede referirse a una clase de cuyas instanciaciones se conciben para notificar al equilibrador 315 de carga que la GC instanciada mediante MaxGenerationNotifier se ha finalizado. Por lo tanto, una instanciación de MaxGenerationCompleteNotifier puede servir para notificar al equilibrador 315 de carga que la redistribución de una o más porciones de la aplicación ejecutable, programa, procedimiento, función u otro conjunto de código puede cesar sin afectar negativamente la ejecución por el entorno 200 de ejecución de tiempo de ejecución. Más particularmente, la MaxGenerationNotifier puede instanciarse cuando el gestor 210 de memoria o administrador 215, ya sea singularmente o combinado, detecta al menos uno de: finalización de la GC que ha servido como catalizador de una instanciación más reciente de MaxGenerationNotifier; un umbral predeterminado de tiempo ha pasado desde la instanciación de MaxGenerationNotifier; u otro módulo programado de código se ha ejecutado para la aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución.

El equilibrador 315 de carga puede referirse a cualquiera de un dispositivo de hardware que está físicamente separado de cualquiera de dispositivo 105 de cliente o como componente de firmware o software en al menos uno de dispositivo 105 de cliente o dispositivos 110A y 110B de servidor.

Tras una instanciación de MaxGenerationNotifier, el equilibrador 315 de carga puede redistribuir el procesamiento de al menos porciones de una aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución para evitar una pausa durante GC. Por ejemplo, tras la recepción de datos 310 de gestión de memoria que incluyen una instanciación de MaxGenerationNotifier haciendo referencia a una GC inminente en procedimiento 305A, el equilibrador 315 de carga puede recibir una o más solicitudes para ejecución de porciones de la misma u otra aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución, y redirigir tal ejecución a al menos uno de procedimiento 305B y procedimiento 305C.

Tras una instanciación de MaxGenerationCompleteNotifier, el equilibrador 315 de carga puede cesar la redistribución de procesamiento de porciones de una aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución a procedimiento 305B y procedimiento 305C. Por lo tanto, el equilibrador 315 de carga puede tener la opción de tener el equilibrio de la aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución ejecutado en el procedimiento 305A.

Al menos una implementación alternativa de flujo 300 de datos no incluye equilibrador 315 de carga. Es decir, tales

implementaciones alternativas logran el equilibrio de carga teniendo los procedimientos 305A, 305B y 305C

La Figura 4 muestra flujo 400 de procesamiento de ejemplo que corresponde a al menos una implementación de ejemplo de equilibrio 120 de carga (véase la Figura 1). El flujo 400 de procesamiento de ejemplo se describe a continuación con referencias a prestaciones de las Figuras 1-3, aunque tales implementaciones se proporcionan únicamente como ejemplos y no se pretende que se interpreten de ninguna manera limitante.

El bloque 405 puede referirse a ejecución de al menos una porción de una aplicación, programa, procedimiento, función u otro conjunto de código que se implementa como parte del procedimiento 305A, procedimiento 305B o procedimiento 305C, ya sea singularmente o colaborativamente en diversas combinaciones. Aunque el equilibrio 120 de carga no se limita de tal forma, el bloque 405 puede referirse a ejecución en un entorno de ejecución gestionado (es decir, entorno 200 de ejecución de tiempo de ejecución).

El bloque 410 puede referirse al gestor 210 de memoria o administrador 215, ya sea singularmente o en combinación, determinando que una cantidad umbral de memoria dinámica ya se ha asignado durante ejecución de una aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución y por lo tanto una GC puede ser inminente. Como alternativa, el bloque 410 puede referirse al gestor 210 de memoria o administrador 215, ya sea singularmente o en combinación, inspeccionando al menos una porción del código programado para determinar que puede ser inminente un barrido o exploración de una memoria dinámica del dispositivo de procesamiento en el que se ejecuta el entorno 200 de ejecución de tiempo de ejecución.

La decisión 415 puede referirse al gestor 210 de memoria o administrador 215, ya sea singularmente o en combinación, evaluando la magnitud de la implementación de GC anticipada en relación a un valor umbral predeterminado. La evaluación puede hacerse para determinar si la magnitud física de la memoria dinámica excede un valor umbral predeterminado (por ejemplo, 2GB o más) que probablemente pausaría la ejecución de una aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución. De acuerdo con una alternativa, la evaluación puede hacerse del tiempo anticipado para que la GC finalice contra un umbral de tiempo predeterminado para evitar una pausa. De acuerdo con otro ejemplo más, la evaluación puede incluso incluir inspeccionar al menos una porción del código programado para determinar si la GC inminente se pretende deliberadamente que sirva como un catalizador para al menos porciones de la aplicación, programa, procedimiento, función u otro conjunto de código en ejecución a redistribuirse.

El bloque 420A, posterior a la decisión 415 negativa, puede referirse al gestor 210 de memoria implementando GC para la aplicación, programa, procedimiento, función u otro conjunto de código en ejecución en el anfitrión uno de procedimientos 305A, 305B o 305C. El flujo de procesamiento puede entonces retornar al bloque 405 para continuar la ejecución de la aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución.

El bloque 425, posterior a la decisión 415 positiva, puede referirse al gestor 210 de memoria o administrador 215, ya sea singularmente o en combinación, notificando al equilibrador 315 de carga que la GC anticipada cumple o excede los valores umbral predeterminados. Es decir, tras la determinación de que un parámetro lógico o físico de la GC anticipada se espera que exceda un valor umbral, el módulo 210 de gestión o administrador 215, ya sea singularmente o en combinación, puede notificar al equilibrador de carga que la GC inminente es probable que provoque una pausa para una aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución. Incluso más particularmente, el módulo 210 de gestión o administrador 215, ya sea singularmente o en combinación, puede instanciar una API (por ejemplo, MaxGenerationNotifier), que proporciona una notificación de estado anticipado.

En al menos una implementación alternativa de bloque 425, el gestor 210 de memoria o administrador 215, ya sea singularmente o en combinación, puede proporcionar una notificación de otra forma. Por ejemplo, la notificación puede ser un evento que incluye llamar a la aplicación, programa, procedimiento, función u otro conjunto de código que se ejecuta como parte del procedimiento 305A, procedimiento 305B o procedimiento 305C anteriormente registrado con el gestor 210 de memoria o administrador 215.

Un ejemplo de un evento de notificación de este tipo puede incluir cada uno de procedimientos 305A, 305B y 305C en la Figura 3 suscribiéndose a uno o más eventos de gestión de memoria (por ejemplo, MaxGenerationNotifier) a través del gestor 210 de memoria o administrador 215 ya sea singularmente o combinado. Por ejemplo, el procedimiento 305A puede implementar dos funciones que pueden llamarse cuando MaxGenNotifier se emite por el gestor 210 de memoria o administrador 215 ya sea singularmente o en combinación. Ejemplos no limitantes de tales llamadas son como siguen:

(1) GC.MaxGenerationNotifier += new MaxGenerationNotificationHandler(On_MaxGeneration_Notify1);
(2) GC.MaxGenerationNotifier += new MaxGenerationNotificationHandler(On_MaxGeneration_Notify2);

con lo que MaxGeneration_Notify1 y MaxGeneration_Notify2 son las funciones anteriormente mencionadas. Además, las funciones MaxGeneration_Notify1 y MaxGeneration_Notify2 pueden notificar al equilibrador 315 de carga de los eventos suscritos. Por lo tanto, tras el suceso de un evento suscrito, el equilibrador 315 de carga puede tomar acción de equilibrio de carga solicitada.

El bloque 430 puede referirse al equilibrador 315 de carga redistribuyendo al menos porciones de la aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución entre los procedimientos 305A, 305B y 305C. Por lo tanto, por ejemplo, si la notificación en el bloque 425 se recibe en anticipación de una implementación de GC de acuerdo con el procedimiento 305A de anfitrión, el bloque 430 puede referirse al equilibrador 315 de carga recibiendo una solicitud para ejecución de al menos una porción de la misma u otra aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución y redirigir la ejecución solicitada a al menos uno de procedimientos 305B y 305C. Es decir, el bloque 430 puede referirse al equilibrador 315 de carga redistribuyendo al menos porciones de la aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución del procedimiento 305A actualmente en ejecución a uno en ejecución simultáneamente de los procedimientos 305B y 305C.

El bloque 420B puede referirse al gestor 210 de memoria implementando GC para la aplicación, programa, procedimiento, función u otro conjunto de código en ejecución en el anfitrión uno de procedimientos 305A, 305B o 305C, mientras al menos porciones de la respectiva aplicación, programa, procedimiento, función u otro conjunto de código se están ejecutando en otros de procedimientos 305A, 305B y 305C en ejecución simultáneamente. Por lo tanto, por ejemplo, si la notificación en el bloque 425 se recibe en anticipación de una implementación de GC de acuerdo con el procedimiento 305A de anfitrión, el bloque 430 puede referirse al equilibrador 315 de carga redistribuyendo al menos porciones de la aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución a al menos uno de procedimientos 305B y 305C y el bloque 420A puede referirse al gestor 210 de memoria implementando GC en el procedimiento 305A de anfitrión.

El bloque 435 puede referirse al gestor 210 de memoria o administrador 215, ya sea singularmente o en combinación, notificando al equilibrador 315 de carga que la GC se ha finalizado. La notificación puede o no puede incluir instrucciones para cesar una implementación de equilibrio de carga actual. Incluso más particularmente, el módulo 210 de gestión o administrador 215, ya sea singularmente o en combinación, puede instanciar una API (por ejemplo, MaxGenerationNotifier), que proporciona una notificación de estado actual. El flujo de procesamiento puede entonces retornar al bloque 405 para continuar la ejecución de una aplicación, programa, procedimiento, función u otro conjunto de código actualmente en ejecución.

Similar al procesamiento en el bloque 425, un evento de notificación puede incluir cada uno de procedimientos 305A, 305B y 305C en la Figura 3 suscribiendo al uno o más eventos de gestión de memoria (por ejemplo, MaxGenerationCompleteNotifier) a través de gestor 210 de memoria o administrador 215 ya sea singularmente o combinado. Por lo tanto, por ejemplo, procedimiento 305A puede implementar dos funciones que pueden llamarse cuando MaxGenCompleteNotifier se emite por el gestor 210 de memoria o administrador 215 ya sea singularmente o en combinación. Ejemplos no limitantes de tales llamadas son como siguen:

```
(1) GC.MaxGenerationCompleteNotifier += new MaxGenerationCompleteNotificationHandler(On_MaxGenerationComplete_Notify1);
(2) GC.MaxGenerationCompleteNotifier += new MaxGenerationCompleteNotificationHandler(On_MaxGenerationComplete_Notify2);
```

con lo que MaxGenerationComplete_Notify1 y MaxGenerationComplete_Notify2 son las funciones anteriormente mencionadas. Además, las funciones MaxGenerationComplete_Notify1 y MaxGenerationComplete_Notify2 pueden notificar al equilibrador 315 de carga de los eventos suscritos. Por lo tanto, tras suceso de un evento suscrito, el equilibrador 315 de carga puede cesar al menos temporalmente equilibrio de carga, según se solicite.

La descripción anterior, perteneciente a las Figuras 1-4, equilibrio de carga con respecto a una o más aplicaciones ejecutándose en un entorno de ejecución de tiempo de ejecución puede implementarse de acuerdo con datos asociados con datos de gestión de memoria. Sin embargo, las limitaciones de ejemplo descritas en el presente documento no se limitan a solo equilibrio de carga a base de datos de gestión de memoria. En su lugar, una notificación (es decir, notificación) puede instanciarse para implementar equilibrio de carga a base de numerosos criterios lógicos y físicos.

El entorno informático para cualquiera de los ejemplos e implementaciones descrito anteriormente puede incluir un dispositivo informático que tiene, por ejemplo, uno o más procesadores o unidades de procesamiento, una memoria de sistema y un bus de sistema para acoplar diversos componentes de sistemas.

El dispositivo informático puede incluir una diversidad de medio legible por ordenador, incluyendo tanto medio volátil como no volátil, memoria extraíble y no extraíble. La memoria de sistema puede incluir medio legible por ordenador en forma de memoria volátil, tal como memoria de acceso aleatorio (RAM); y/o memoria no volátil, tal como memoria de solo lectura (ROM) o RAM flash. Se aprecia que otros tipos de medio legible por ordenador que pueden almacenar datos que son accesibles mediante un ordenador, tales como cintas magnéticas u otros dispositivos de almacenamiento magnético, tarjetas de memoria flash, CD-ROM, discos versátiles digitales (DVD) u otro almacenamiento óptico, memorias de acceso aleatorio (RAM), memorias de solo lectura (ROM), memoria de solo lectura eléctricamente programable y borrable (EEPROM) y similares, también pueden utilizarse para implementar el sistema informático y entorno de ejemplo.

Se ha hecho referencia por toda esta memoria descriptiva a "un ejemplo," " ejemplos alternativos," "al menos un ejemplo," "una implementación," o "una implementación de ejemplo" significando que una prestación, estructura o característica se incluye en al menos una implementación de la presente invención. Por lo tanto, el uso de tales frases puede referirse a más de solo una implementación. Adicionalmente, las prestaciones, estructuras o características descritas pueden combinarse de cualquier manera adecuada en una o más implementaciones.

Un experto en la técnica relevante puede reconocer, sin embargo, que la invención puede practicarse sin uno o más de los detalles específicos o con otros procedimientos, recursos, materiales, etc. En otros casos, estructuras, recursos u operaciones bien conocidas no se ha mostrado o descrito en detalle meramente para evitar obstaculizar aspectos de la invención.

REIVINDICACIONES

1. Un procedimiento de equilibrio de carga en un entorno informático, comprendiendo dicho entorno informático un gestor (210, 310) de memoria dentro de un entorno (200) de ejecución gestionado, siendo el gestor de memoria capaz de iniciar el equilibrio de carga entre procedimientos (305A, 305B, 305C) actualmente en ejecución pertenecientes a límites de aislamiento, a través de sus límites de aislamiento, comprendiendo dichos límites de aislamiento límites de máquina y/o límites de procedimiento, en el que los procedimientos se refieren a una ejecución actual de un código perteneciente a un programa, a continuación denominado como código, comprendiendo el procedimiento las etapas de:

recibir (425) en un equilibrador (315) de carga una indicación de que la asignación de memoria durante la ejecución (405) de código en el entorno (200) de ejecución gestionado excede un valor umbral, detectando de este modo una recolección de basura inminente; y
redistribuir (430) mediante el equilibrador (315) de carga la ejecución de al menos una porción del código desde un primer procedimiento (305A) actualmente en ejecución de los procedimientos actualmente en ejecución a al menos uno de los otros procedimientos (305B,305C) actualmente en ejecución a través de un límite de aislamiento, evitando de este modo la suspensión de la ejecución de código en vista de la recolección de basura que se implementa en el primer procedimiento (305A) actualmente en ejecución.

2. El procedimiento de acuerdo con la reivindicación 1, en el que el procedimiento se realiza de acuerdo con un equilibrador (315) de carga.

3. El procedimiento de acuerdo con la reivindicación 1, en el que la indicación recibida incluye el aviso para preparar una pausa de ejecución anticipada.

4. El procedimiento de acuerdo con la reivindicación 1, en el que la indicación recibida incluye una notificación de que se espera que un barrido inminente de memoria dinámica provoque una pausa de ejecución.

5. El procedimiento de acuerdo con la reivindicación 1, en el que la indicación recibida incluye una vista previa de estado en conexión con la ejecución del código.

6. El procedimiento de acuerdo con la reivindicación 1, en el que la recepción adicionalmente incluye recibir una instanciación de una interfaz de programación de aplicación, API, de un componente de gestión de memoria asociado con el entorno de ejecución gestionado.

7. El procedimiento de acuerdo con la reivindicación 1, en el que el límite de aislamiento es un límite de máquina o un límite de procedimiento.

8. El procedimiento de acuerdo con la reivindicación 1, en el que la redistribución incluye redistribuir la ejecución de al menos una porción de la aplicación.

9. El procedimiento de acuerdo con la reivindicación 1, en el que la redistribución incluye redirigir solicitudes para ejecución de al menos porciones de la aplicación a través del límite de aislamiento.

10. El procedimiento de acuerdo con la reivindicación 1, en el que la redistribución incluye redirigir una solicitud para ejecución de una porción de la aplicación a otro servidor.

11. Al menos un medio legible por ordenador que tiene una o más instrucciones ejecutables que, cuando se leen, provocan que uno o más procesadores realicen todas las etapas de procedimiento de una de las reivindicaciones 1 a 10.

12. Un sistema de equilibrio de carga en un entorno informático, comprendiendo dicho entorno informático un gestor (210) de memoria dentro de un entorno de ejecución gestionado, siendo el gestor de memoria capaz de iniciar el equilibrio de carga entre procedimientos (305A, 305B, 305C), pertenecientes a límites de aislamiento, a través de sus límites de aislamiento, comprendiendo dichos límites de aislamiento límites de máquina y/o límites de procedimiento, en el que los procedimientos se refieren a la ejecución actual de un módulo o conjunto de código perteneciente al procedimiento, a continuación denominado como código, estando el sistema adaptado para realizar al menos uno de los procedimientos de las reivindicaciones 1 a 10.

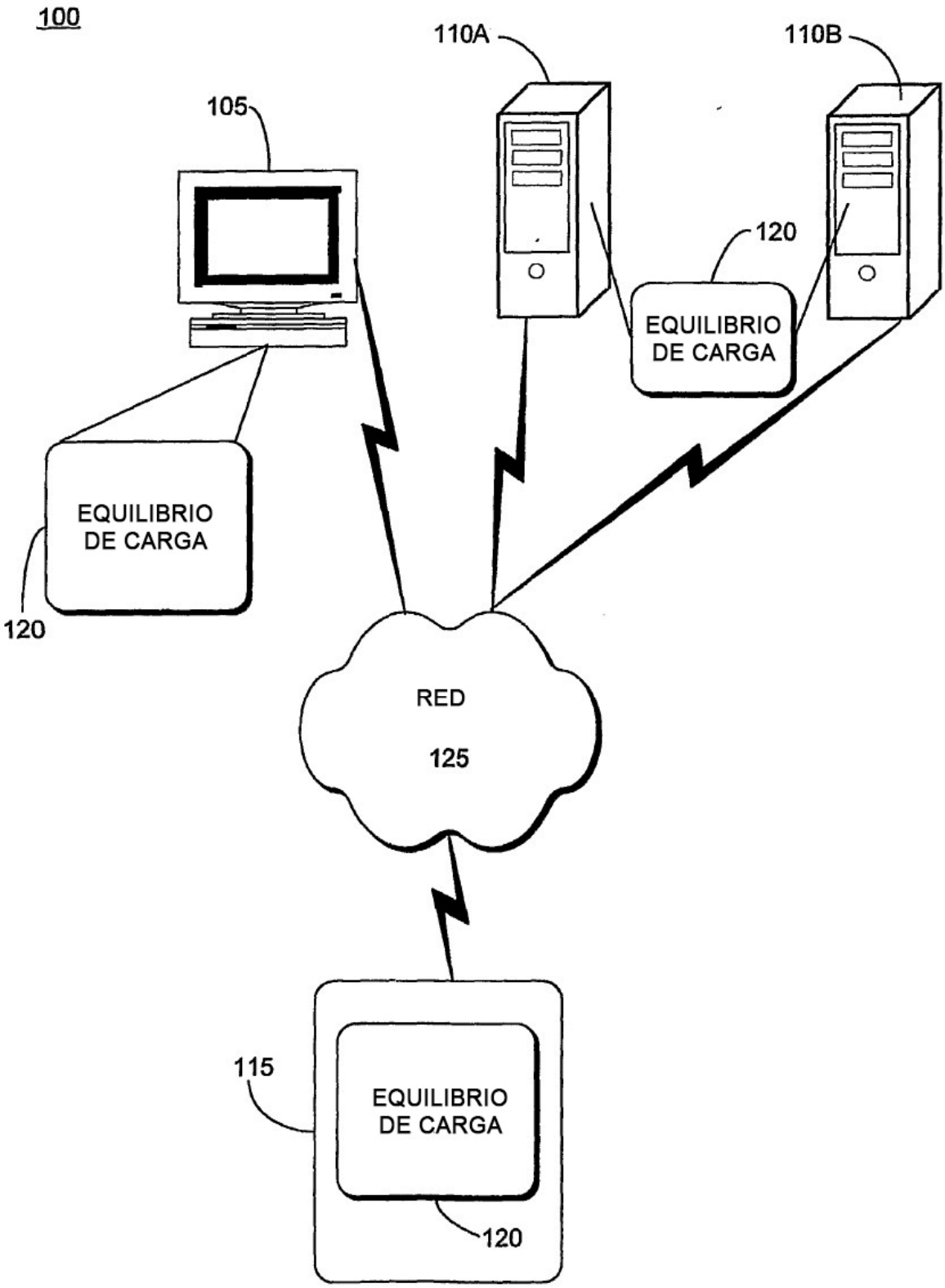
100
110A
110B
105
120
EQUILIBRIO DE CARGA
EQUILIBRIO DE CARGA
120
RED
125
115
EQUILIBRIO DE CARGA
120

FIG. 1

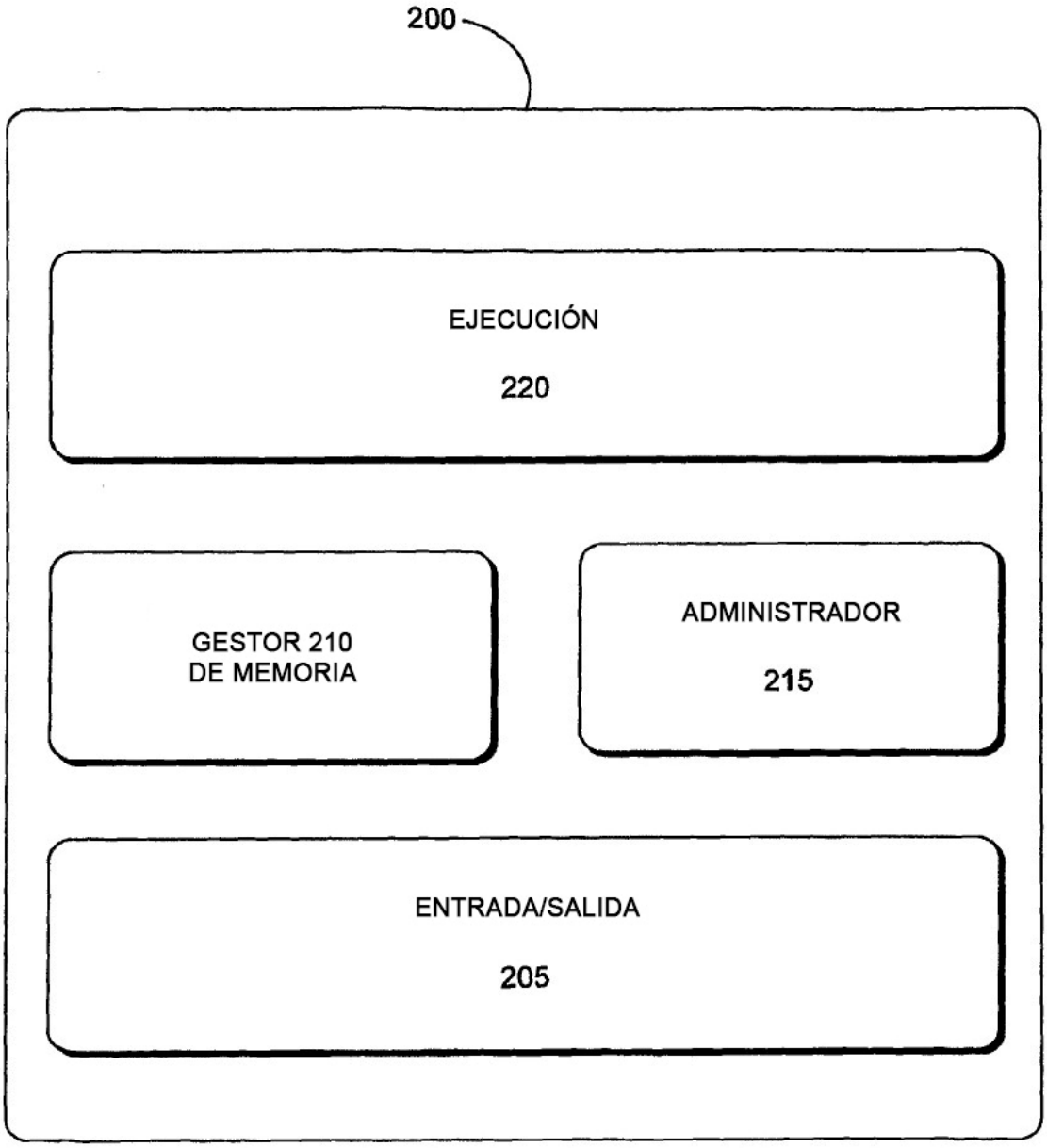
200
EJECUCIÓN
220
GESTOR 210
DE MEMORIA
ADMINISTRADOR
215
ENTRADA/SALIDA
205

FIG. 2

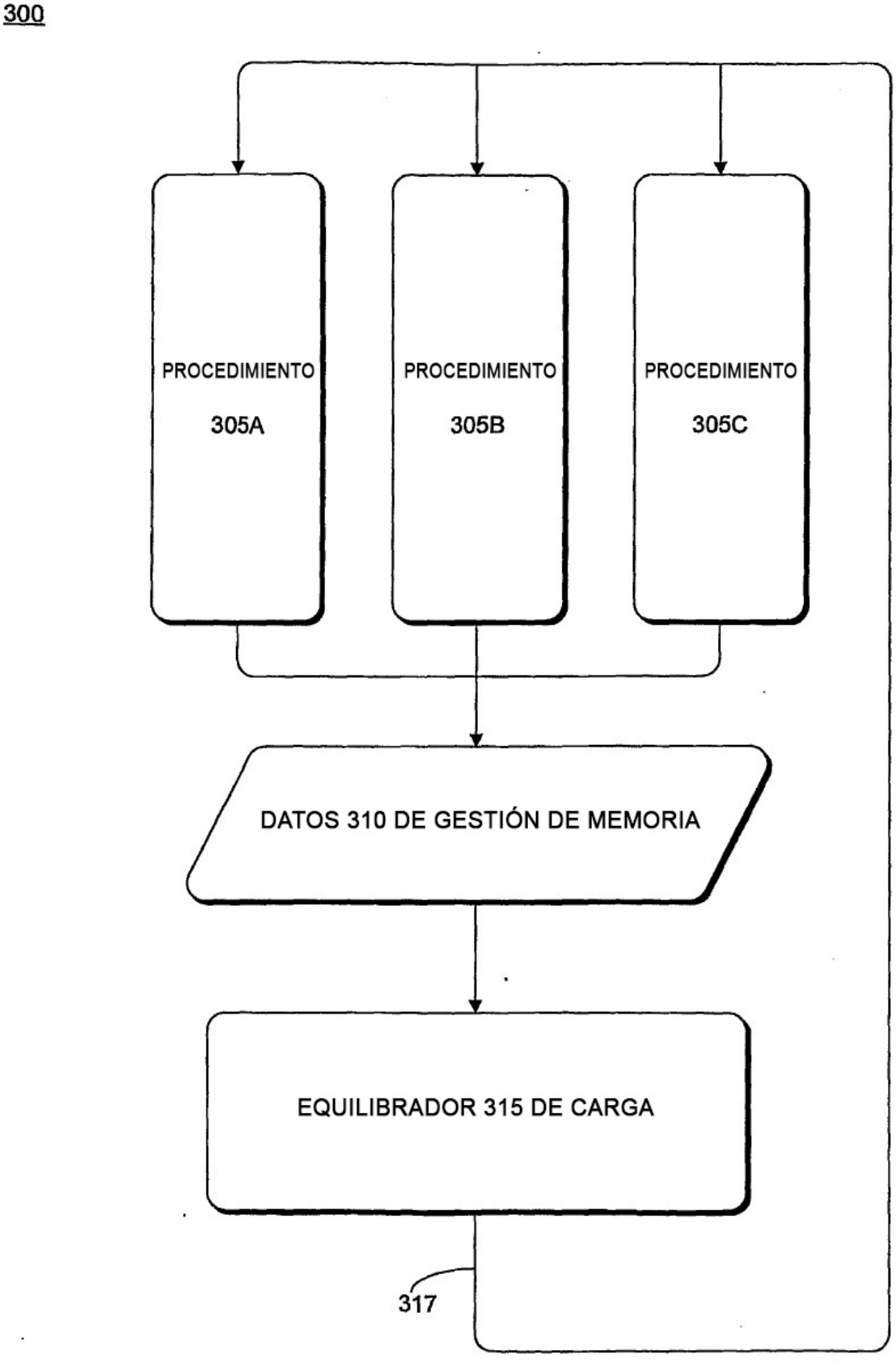
300
PROCEDIMIENTO
305A
PROCEDIMIENTO
305B
PROCEDIMIENTO
305C
DATOS 310 DE GESTIÓN DE MEMORIA
EQUILIBRADOR 315 DE CARGA
317

FIG. 3

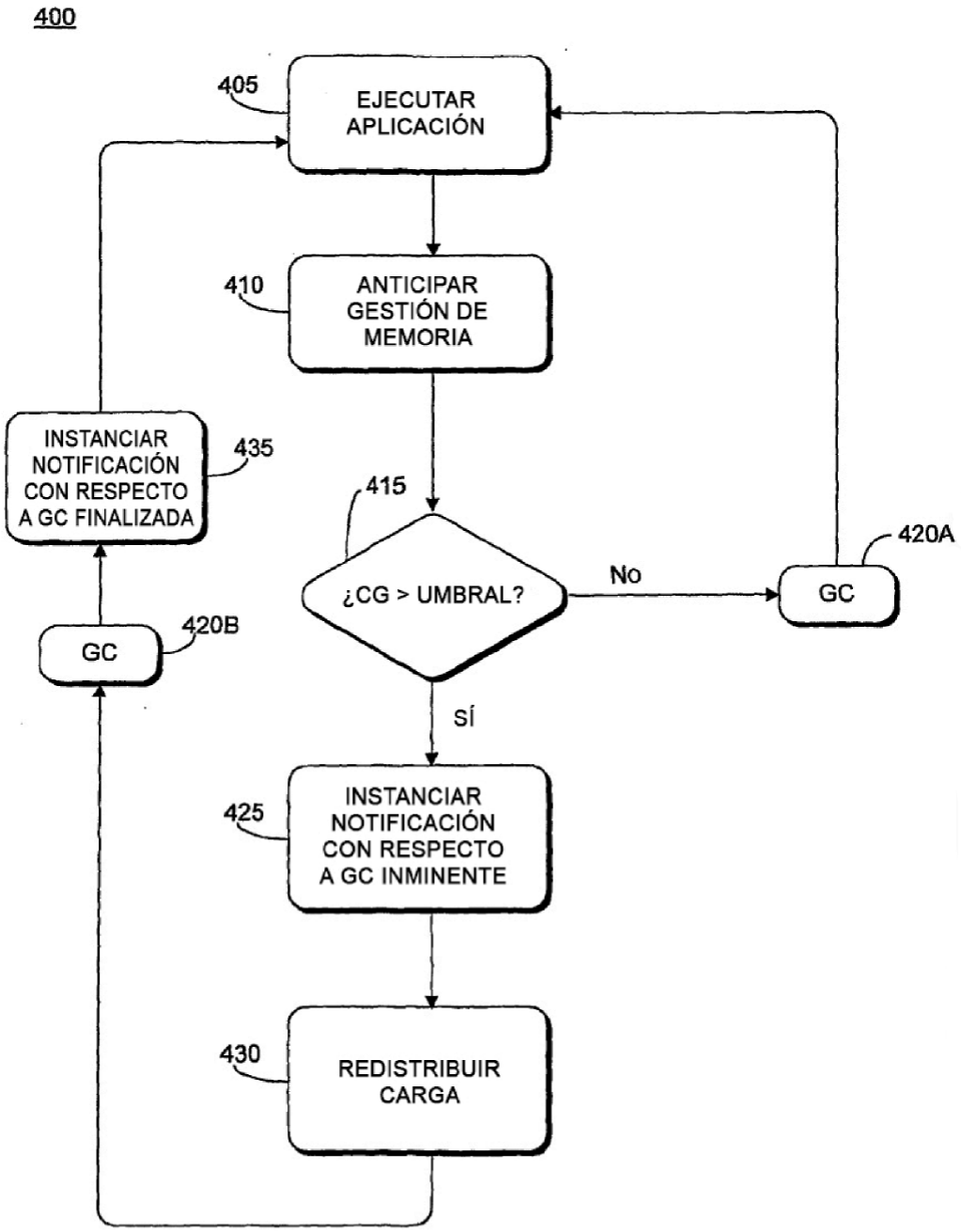
400
405
EJECUTAR APLICACIÓN
410
ANTICIPAR GESTIÓN DE MEMORIA
415
¿CG > UMBRAL?
No
420A
GC
SÍ
425
INSTANCIAR NOTIFICACIÓN CON RESPECTO A GC INMINENTE
430
REDISTRIBUIR CARGA
420B
GC
435
INSTANCIAR NOTIFICACIÓN CON RESPECTO A GC FINALIZADA

FIG. 4