(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 717 603**

(51) Int. CI.:

G06F 9/4401 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **14.04.2005 E 05102941 (1)**

(97) Fecha y número de publicación de la concesión europea: **09.01.2019 EP 1594052**

(54) Título: **VEX - Marco de extensión virtual**

(30) Prioridad:

30.04.2004 US 837971

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
24.06.2019

(73) Titular/es:

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

(72) Inventor/es:

WOBBER, EDWARD P.;
BARHAM, PAUL;
ROEDER, THOMAS y
ERLINGSSON, ULFAR

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

VEX - Marco de extensión virtual

Solicitud relacionada

La presente solicitud está relacionada con la solicitud de EE. UU., en tramitación junto con la presente, titulada "*Providing Direct Access To Hardware From A Virtual Environment*", número de expediente del mandatario 226339, que fue presentada en la misma fecha que la presente solicitud.

Campo de la invención

La presente invención se refiere, en general, a las extensiones tolerantes a fallos y, más en particular, se refiere a un sistema y procedimiento para proporcionar un entorno virtual aislado frente a fallos para las extensiones.

Antecedentes

Un mecanismo cada vez más popular para extender los sistemas operativos o aplicaciones es el uso de extensiones que pueden proporcionar una funcionalidad adicional y se pueden entregar e invocar de una forma individual. Los sistemas operativos modernos, por ejemplo, dependen de extensiones que son proporcionadas por los fabricantes de hardware para la interacción entre los componentes de hardware que son realizados por los fabricantes de hardware y el software de sistema operativo. De tal forma, un autor de sistemas operativos solo necesita proporcionar un soporte de hardware general, y no necesita intentar soportar cada dispositivo de hardware concebible. Cuando se añade un nuevo dispositivo de hardware a un sistema informático en el que se está ejecutando el sistema operativo, una extensión que se diseña de forma específica para interaccionar entre el nuevo dispositivo de hardware y el sistema operativo puede ser proporcionada por el fabricante de hardware y puede ser usada por el sistema operativo para controlar el nuevo dispositivo de hardware y proporcionar al usuario un acceso a la funcionalidad del nuevo dispositivo de hardware.

Además del sistema operativo, muchas otras aplicaciones de software dependen de extensiones para proporcionar una funcionalidad adicional al tiempo que se reduce la complejidad de la aplicación de host. Por ejemplo, una aplicación de navegador web puede depender de extensiones para proporcionar a los usuarios la capacidad de interpretar o interaccionar con una amplia diversidad de tipos de datos. Por lo tanto, un navegador web solo necesita proporcionar la capacidad de manejar unos tipos limitados de datos, tales como las páginas web que se escriben en el Lenguaje de Marcado de HiperTexto (HTML, *HyperText Markup Language*) o unas imágenes que se codifican usando el formato de codificación del Grupo Conjunto de Expertos en Fotografía (JPEG, *Joint Photographic Experts Group*). Las páginas web que requieren unas capacidades adicionales de la aplicación de navegador pueden depender de unas extensiones personalizadas para que la aplicación de navegador proporcione las capacidades requeridas. Por ejemplo, un autor de páginas web que desea usar unas imágenes que se codifican en un formato poco usado puede proporcionar una extensión que puede entender el formato de imagen particular y puede interaccionar con el navegador web para habilitar que el navegador web represente unas imágenes que se codifican usando ese formato.

Otras aplicaciones también pueden usar extensiones para proporcionar una funcionalidad aumentada. Por ejemplo, las aplicaciones de procesamiento de imágenes pueden usar extensiones que son proporcionadas por una diversidad de artistas para permitir que los usuarios accedan a muchos algoritmos de procesamiento de imágenes artísticas diferentes que son desarrollados por esos artistas. Un programa de procesamiento de imágenes puede, por lo tanto, proporcionar unas características básicas de edición de imágenes, tales como unos controles de color y de contraste. Unas características más complejas, tales como un efecto que parece transformar una fotografía en una pintura al óleo, pueden ser proporcionadas por extensiones que se encuentran disponibles por separado. De tal forma, los usuarios de las aplicaciones de procesamiento de imágenes pueden seleccionar de forma individual qué extensiones son las más apropiadas para sus requisitos y pueden elegir instalar solo aquellas extensiones que pueden ser útiles dados esos requisitos.

En general, las extensiones interaccionan con las aplicaciones de software de host a través de diversas interfaces de programa de aplicaciones (API, *application program interface*). Una API define la forma en la que se va a acceder a diversas características o capacidades. Por lo tanto, se dice que dos componentes interaccionan a través de una API cuando un componente comprende un código y unos datos para poner en práctica una función particular y prevé la función se invoque de la forma que se define por medio de la API, y un segundo componente solicita la función de la forma que se define por medio de la API.

En general, la interacción entre una extensión y una aplicación de software de host tiene lugar a través de dos tipos de API: las API de servicio y las API de soporte. Las API de soporte se pueden poner en práctica por medio de la aplicación de software de host, el sistema operativo, o cualquier otro software que pueda ser usado por la extensión para acceder a la funcionalidad que es proporcionada por la aplicación de software de host. Además, las API de soporte se pueden disponer en capas, o apilar, de tal modo que algunas API de soporte pueden depender de unas API de soporte adicionales para llevar a cabo las funciones solicitadas. Las API de servicio, por otro lado, se pueden poner en práctica por medio de la extensión y pueden ser usadas por la aplicación de software de host para acceder

a la funcionalidad que es proporcionada por la extensión. Como un ejemplo, una extensión a un navegador web que habilita que el navegador web visualice unas imágenes que se codifican en un formato particular puede exponer las API de servicio que el navegador web puede usar para pasar los datos de imagen a la extensión y solicitar que la extensión interprete los datos de imagen, mientras que el navegador web puede proporcionar unas API de soporte que puede proporcionar unos servicios básicos que la extensión puede solicitar al navegador web.

Muchas extensiones, en especial las extensiones de sistema operativo, tales como las que se diseñan para interaccionar con los dispositivos de hardware, pueden depender de múltiples API de soporte para interaccionar de forma apropiada con el hardware. Muchos dispositivos de hardware que se pueden añadir a un dispositivo informático moderno interaccionan físicamente con el dispositivo informático a través de unos puertos que se controlan por medio de un conjunto de circuitos o unidades de procesamiento dedicadas. Las API de soporte del sistema operativo puede proporcionar mecanismos por medio de los cuales el conjunto de circuitos dedicados puede ser usado por una extensión con el fin de que la extensión acceda a, y controle, el dispositivo de hardware para el cual se diseñó la misma. Por ejemplo, un sistema operativo puede proporcionar unas API de soporte que permiten que las extensiones accedan a unas áreas específicas de memoria de acceso aleatorio que se dejan al margen para las comunicaciones de entrada / salida de hardware. Como alternativa, un sistema operativo puede proporcionar unas API de soporte que permiten que las extensiones accedan a los puertos de entrada / salida de hardware directamente.

Las extensiones, por otro lado, pueden proporcionar unas API de servicio para permitir que el sistema operativo u otras aplicaciones de software accedan a la funcionalidad que es proporcionada por la extensión. Por ejemplo, un controlador de dispositivo para un nuevo dispositivo de representación puede comprender un código y unos datos para controlar el dispositivo de representación. Ese código y esos datos del controlador de dispositivo de representación se pueden invocar por medio de otro código, tal como un código de sistema operativo o un código de aplicación de software de una forma previamente definida a través del uso de las API de servicio. En el ejemplo de un controlador de dispositivo de representación, las API de servicio pueden incluir unas API para solicitar la representación de datos de imagen, unas API para solicitar información a partir del sistema de representación, tal como su resolución actual, y unas API para solicitar la activación de las características de ahorro de energía del sistema de representación.

Desafortunadamente, debido a que en general las aplicaciones de software de host y las extensiones son creadas por unos individuos diferentes, puede que sea común que la extensión no interaccione de forma correcta con el programa de host. Por ejemplo, algunas extensiones pueden no usar de forma correcta las API de soporte que son proporcionadas por la aplicación de software de host, pasando parámetros o datos no apropiados o, por lo demás, solicitando de forma no apropiada diversas funciones. Otras extensiones pueden intentar sortear las API de soporte en su totalidad y acceder a la funcionalidad directamente en lugar de a través de la aplicación de software de host, tal como un acceso de disco, un acceso por pantalla, y similares. Como alternativa, las extensiones también pueden intentar acceder a las API de soporte que no tenían por objeto su uso por las extensiones, tales como unas API que no están plenamente documentadas o tenían por objeto solo un uso interno dentro de la propia aplicación de software.

Una extensión que no interacciona de forma correcta con su aplicación de software de host puede dar como resultado un fallo de la aplicación de software de host global, o incluso la totalidad del sistema informático. Esto puede ser especialmente problemático cuando el usuario de la aplicación de software de host se ve forzado a depender de múltiples extensiones a partir de múltiples entidades de forma simultánea, tales como la muy grande e indefinida cantidad de controladores de dispositivo de terceras partes que comúnmente son usados por un sistema operativo moderno. Además, a menudo es difícil el diagnóstico de tales fallos y el darles soporte de forma correcta debido a que, a pesar de que la aplicación de software de host pueda ser culpada por el usuario, de hecho es la extensión la que es la causa de la inestabilidad. Por lo tanto, es deseable crear un entorno de extensión que pueda proporcionar un acceso a las características y capacidades de la extensión a la aplicación de software de host al tiempo que se aísla la aplicación de software de host con respecto a la inestabilidad que se pueda introducir por medio de la extensión.

El documento de patente US6546431 divulga un sistema de procesamiento de datos para compartir dispositivos de interfaz de usuario tales como lectores de pantalla o dispositivos de salida en Braille de aplicaciones de tecnología de ayuda diferentes para un ordenador tal como un ordenador personal, un miniordenador, un ordenador de gran sistema y un ordenador que está conectado con una red informática distribuida y que es operada por usuarios con minusvalías tales como usuarios ciegos. Este permite que los lectores de pantalla que se están ejecutando en el host y en la máquina virtual cooperen de forma perfecta entre sí, permite que un lector de pantalla de Java que se está ejecutando en la máquina virtual use los objetos de entrada / salida que están conectados con el gestor de entrada / salida y reduce al mínimo la codificación que es necesaria para permitir la cooperación entre los lectores de pantalla.

Breve sumario de la invención

Las formas de realización de la invención permiten que las extensiones se ejecuten en un entorno protegido a partir del cual las mismas pueden proporcionar los beneficios previstos a la aplicación de software de host al tiempo que

se aísla la aplicación de host con respecto a la inestabilidad a la que da lugar la extensión.

En una forma de realización, una extensión se puede ejecutar en un entorno virtual que puede proporcionar las API de soporte que la extensión puede requerir de forma correcta al tiempo que se aísla la extensión con respecto a la aplicación de software de host.

En otra forma de realización, una extensión se puede ejecutar en un entorno virtual y una extensión de entidad representante correspondiente se puede ejecutar en el espacio de proceso de la aplicación de software de host de una forma tal que la extensión de entidad representante proporciona un acceso a las capacidades de la extensión original al tiempo que se aísla la aplicación de software de host con respecto a la extensión original a través del uso del entorno virtual. La extensión y la entidad representante pueden ser similares, con el fin de proporcionar un aislamiento frente a fallos entre una aplicación de software de host y una extensión que se diseña para interaccionar con la aplicación de software de host, o la extensión y la entidad representante pueden ser diferentes, con el fin no solo proporcionar un aislamiento frente a fallos, sino también de ampliar la funcionalidad de una aplicación de software de host al habilitar que la aplicación de software de host utilice una extensión que puede no haber sido diseñada para interaccionar con la aplicación de software de host.

En una forma de realización adicional, las extensiones que pueden requerir un contexto de modo de usuario se pueden ejecutar en un entorno virtual mediante la creación de una aplicación de software de host suplente en el entorno virtual, o al cambiar la asignación de tabla de páginas en el entorno virtual de tal modo que la gama de modos de usuario se asigna a la misma memoria física tanto en el entorno virtual como en el entorno de host o, como aún otra alternativa, el contexto de modo de usuario se puede copiar en el entorno virtual. Para mantener el aislamiento frente a fallos, se puede acceder a la memoria física del entorno de host de una forma de tipo solo lectura, o mediante el uso de técnicas de copia en escritura.

En una forma de realización adicional, un entorno virtual en el que se pueden ejecutar de forma segura unas extensiones se puede crear de forma eficiente al arrancar un entorno virtual original que tiene la funcionalidad, o un superconjunto de la funcionalidad, que es requerida por la extensión, y entonces, en un punto posterior en el tiempo en el que se requiere un entorno virtual, clonar el estado del entorno virtual arrancado original para crear el entorno virtual requerido.

En aún otra forma de realización, un entorno virtual en el que se pueden ejecutar de forma segura unas extensiones se puede crear de forma eficiente durante la secuencia de inicio inicial de un entorno de host al indicar al entorno de host que una segunda unidad de procesamiento se encuentra presente en el sistema informático y entonces, después de que el entorno de host haya completado la creación de un estado coherente para la segunda unidad de procesamiento, indicar al entorno de host que ha fallado la segunda unidad de procesamiento y, posteriormente, indicar al entorno virtual que la segunda unidad de procesamiento se encuentra presente e iniciar el entorno virtual usando el estado coherente para la segunda unidad de procesamiento que se crea durante el inicio del entorno de host.

En aún otra forma de realización, un entorno virtual en el que se pueden ejecutar de forma segura unas extensiones que se diseñan para controlar los dispositivos de hardware se puede crear de forma eficiente durante la secuencia de inicio inicial de un entorno de host al indicar al entorno de host que una segunda unidad de procesamiento se encuentra presente en el sistema informático, permitir que el entorno de host cree un estado coherente para la segunda unidad de procesamiento, y permitir que el entorno de host enlace la segunda unidad de procesamiento con el dispositivo de hardware para controlar el cual no está diseñada la extensión. Después de que el entorno de host haya completado la creación de un estado coherente para la segunda unidad de procesamiento, y el enlace con el dispositivo de hardware apropiado, el entorno de host puede recibir una indicación de que la segunda unidad de procesamiento ha fallado mientras que el entorno virtual se puede iniciar usando el estado coherente de la segunda unidad de procesamiento que se crea durante el inicio del entorno de host, proporcionando al entorno virtual, y a cualquier extensión que se esté ejecutando en el mismo, un acceso simplificado al dispositivo de hardware que está enlazado con la segunda unidad de procesamiento.

Algunas características y ventajas adicionales de la invención se volverán evidentes a partir de la siguiente descripción detallada de algunas formas de realización ilustrativas que procede con referencia a las figuras adjuntas.

Breve descripción de los dibujos

A pesar de que las reivindicaciones adjuntas exponen las características de la presente invención de forma particular, la invención, junto con sus objetos y ventajas, se puede entender del mejor modo a partir de la siguiente descripción detallada, tomada junto con los dibujos adjuntos, en los que:

la figura 1 es un diagrama de bloques que ilustra, en general, una arquitectura de dispositivo a modo de ejemplo en la que se pueden poner en práctica algunas formas de realización de la presente invención;

la figura 2 es un diagrama de bloques que ilustra, en general, un entorno a modo de ejemplo para aislar extensiones de acuerdo con algunas formas de realización de la presente invención;

la figura 3 es un diagrama de bloques que ilustra, en general, un acceso a un contexto de modo de usuario de acuerdo con una forma de realización de la presente invención;

la figura 4 es un diagrama de bloques que ilustra, en general, un acceso alternativo a un contexto de modo de usuario de acuerdo con una forma de realización de la presente invención;

la figura 5 es un diagrama de flujo que ilustra, en general, la creación de un estado coherente de acuerdo con una forma de realización de la presente invención;

la figura 6 es un diagrama de flujo que ilustra, en general, una creación alternativa de un estado coherente de acuerdo con una forma de realización de la presente invención; y

la figura 7 es un diagrama de bloques que ilustra, en general, un entorno a modo de ejemplo para proporcionar a las extensiones que se alojan dentro de una máquina virtual un acceso directo al hardware físico de acuerdo con una forma de realización de la presente invención.

Descripción detallada

Muchas aplicaciones de software y sistemas operativos dependen de extensiones para proporcionar una funcionalidad, unos servicios o capacidades adicionales al usuario final. Una extensión que se usa a menudo se conoce como controlador de dispositivo, y puede proporcionar una interfaz entre una aplicación de software de host, que es en general un sistema operativo, y un dispositivo de hardware. Otras extensiones incluyen miniaplicaciones y complementos para las aplicaciones de software de navegador web, filtros, efectos y complementos para las aplicaciones de software de edición de imágenes, y códecs para las aplicaciones de software de audio / vídeo.

Debido a que las extensiones interoperan íntimamente con sus aplicaciones de software de host, la inestabilidad que se introduce por medio de una extensión puede volver inutilizable la totalidad de la aplicación de software de host. En general, las extensiones proporcionan acceso a sus capacidades a través de una o más interfaces de programa de aplicaciones (API, *application program interface*) que pueden ser usadas por la aplicación de software de host. Las API a través de las cuales las extensiones exponen su funcionalidad se denominan en general “API de servicio”. Si la extensión requiere información adicional, recursos adicionales, o similares, la extensión puede solicitar los mismos a partir de la aplicación de software de host a través de una o más API que se denominan en general “API de soporte”. En el caso de que o bien la extensión o bien la aplicación de software de host usara de forma no apropiada las API de soporte o de servicio, o intentara acceder a unas API no documentadas o no soportadas, todo error o artefacto no previsto resultante puede dar lugar a inestabilidad. Debido a que las extensiones operan en general dentro del mismo proceso que su aplicación de software de host, puede ser muy difícil que la aplicación de software de host continúe operando de forma correcta cuando una o más extensiones que se están ejecutando dentro de ese proceso introducen inestabilidad.

Si una extensión se pudiera ejecutar en un proceso separado, de tal modo que toda inestabilidad que se introduce por medio de la extensión se pueda aislar en un proceso que es independiente del proceso de la aplicación de software de host, la aplicación de software de host puede proceder a operar de forma correcta incluso frente a unas extensiones inestables. Para las aplicaciones de software que pueden alojar muchas extensiones, tales como sistemas operativos, aislar cada extensión puede mejorar en gran medida la fiabilidad global del sistema operativo debido a que la posibilidad de fallo aumenta de forma exponencial con cada extensión adicional que se usa. Además, aislar las extensiones permite que los autores de aplicaciones se concentren en la identificación y en la eliminación de las fuentes de inestabilidad dentro de sus propios algoritmos. En consecuencia, algunas formas de realización de la presente invención aíslan las extensiones con respecto a sus aplicaciones de software de host, al tiempo que continúan proporcionando los beneficios de las extensiones a las aplicaciones de software de host.

A pesar de que no se requiere, la invención se describirá en el contexto general de las instrucciones ejecutables por ordenador, tales como módulos de programa, que se están ejecutando por medio de un dispositivo informático. En general, los módulos de programa incluyen rutinas, programas, objetos, componentes, estructuras de datos, y similares que llevan a cabo tareas particulares o ponen en práctica tipos de datos abstractos particulares. En los entornos informáticos distribuidos, las tareas pueden ser llevadas a cabo por medio de unos dispositivos de procesamiento remotos que se enlazan a través de una red de comunicaciones. En un entorno informático distribuido, los módulos de programa se pueden ubicar en unos medios y / o dispositivos de almacenamiento informático tanto locales como remotos. Los expertos en la materia apreciarán que la invención se puede poner en práctica con muchos dispositivos informáticos diferentes, o bien de forma individual o bien como parte de un entorno informático distribuido, en el que tales dispositivos pueden incluir dispositivos de mano, sistemas de múltiples procesadores, electrónica de consumo programable o basada en microprocesadores, PC en red, miniordenadores, ordenadores de gran sistema, y similares.

Pasando a la figura 1, se muestra un dispositivo informático 100 a modo de ejemplo en el que la invención se puede poner en práctica. El dispositivo informático 100 solo es un ejemplo de un dispositivo informático adecuado y no se tiene por objeto sugerir limitación alguna en lo que respecta al ámbito de uso o a la funcionalidad de la invención. Además, el dispositivo informático 100 no se debería interpretar como si tuviera dependencia o requisito alguno en relación con cualquiera o una combinación de los periféricos que se ilustran en la figura 1.

Los componentes del dispositivo informático 100 pueden incluir, pero no se limitan a, una unidad de procesamiento 120, una memoria de sistema 130 y un bus de sistema 121 que acopla diversos componentes de sistema, incluyendo la memoria de sistema a la unidad de procesamiento 120. El bus de sistema 121 puede ser cualquiera de los varios tipos de estructuras de bus, incluyendo un bus de memoria o controlador de memoria, un bus de periféricos y un bus local usando cualquiera de una diversidad de arquitecturas de bus. A modo de ejemplo y no de limitación, tales arquitecturas incluyen un bus de la Arquitectura de Normalización Industrial (ISA, *Industry Standard Architecture*), un bus de la Arquitectura de Micro Canal (MCA, *Micro Channel Architecture*), un bus de ISA Potenciada (EISA, *Enhanced ISA*), un bus local de la Asociación para Normalización de Electrónica y de Vídeo (VESA, *Video Electronics Standards Associate*), y un bus de Interconexión de Componentes Periféricos (PCI, *Peripheral Component Interconnect*), que también se conoce como bus *Mezzanine* (entresuelo). Además, la unidad de procesamiento 120 puede contener uno o más procesadores físicos.

El dispositivo informático 100 incluye, por lo general, una diversidad de medios legibles por ordenador. Los medios legibles por ordenador pueden ser cualesquiera medios disponibles a los que se pueda acceder por medio del dispositivo informático 100 e incluyen unos medios tanto volátiles como no volátiles y tanto extraíbles como no extraíbles. A modo de ejemplo y no de limitación, los medios legibles por ordenador pueden comprender medios de almacenamiento informático y medios de comunicación. Los medios de almacenamiento informático incluyen unos medios tanto volátiles como no volátiles y tanto extraíbles como no extraíbles que se ponen en práctica en cualquier procedimiento o tecnología para el almacenamiento de información, tales como instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos. Los medios de almacenamiento informático incluyen, pero no se limitan a, RAM, ROM, EEPROM, memoria flash u otra tecnología de memoria, CD-ROM, discos versátiles digitales (DVD, *digital versatile disk*) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnética, almacenamientos de disco magnético u otros dispositivos de almacenamiento magnético o cualquier otro medio que se pueda usar para almacenar la información deseada y al cual se pueda acceder por medio del dispositivo informático 100. Los medios de comunicación incorporan, por lo general, instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos en una señal de datos modulada, tal como una onda portadora u otro mecanismo de transporte e incluyen cualquier medio de entrega de información. La expresión "señal de datos modulada" quiere decir una señal que tiene una o más de sus características establecidas o cambiadas de una forma tal que se codifique la información en la señal. A modo de ejemplo y no de limitación, los medios de comunicación incluyen medios cableados, tales como una red cableada o una conexión cableada directa, y medios inalámbricos, tales como medios acústicos, de RF, de infrarrojos y otros medios inalámbricos. Las combinaciones de cualquiera de los anteriores se han incluir dentro del ámbito de los medios legibles por ordenador.

La memoria de sistema 130 incluye unos medios de almacenamiento informático en la forma de una memoria volátil y / o no volátil, tal como una memoria solo de lectura (ROM, *read only memory*) 131 y una memoria de acceso aleatorio (RAM, *random access memory*) 132. Un sistema básico de entrada / salida 133 (BIOS, *basic input / output system*), que contiene las rutinas básicas que ayudan a transferir información entre los elementos dentro del ordenador 110, tales como durante el arranque, por lo general se almacena en la ROM 131. La RAM 132 contiene, por lo general, datos y / o módulos de programa a los que se puede acceder inmediatamente y / o sobre los que se puede estar operando en la actualidad por medio de la unidad de procesamiento 120. A modo de ejemplo y no de limitación, la figura 1 ilustra el sistema operativo 134, los programas de aplicación 135, otros módulos de programa 136 y los datos de programa 137.

El dispositivo informático 100 también puede incluir otros medios de almacenamiento informático extraíbles / no extraíbles y volátiles / no volátiles. Solo a modo de ejemplo, la figura 1 ilustra una unidad de disco duro 141 que lee o escribe en unos medios magnéticos no extraíbles no volátiles, una unidad de disco magnético 151 que lee o escribe en un disco magnético extraíble no volátil 152, y una unidad de disco óptico 155 que lee o escribe en un disco óptico extraíble / volátil 156, tal como un CD-ROM u otros medios ópticos. Otros medios de almacenamiento informático extraíbles / no extraíbles y volátiles / no volátiles que se pueden usar en el sistema operativo a modo de ejemplo incluyen, pero no se limitan a, casetes de cinta magnética, tarjetas de memoria flash, discos versátiles digitales, cintas digitales de vídeo, RAM de estado sólido, memorias ROM de estado sólido, y similares. La unidad de disco duro 141, por lo general, se conecta al bus de sistema 121 a través de una interfaz de memoria no extraíble tal como la interfaz 140 y la unidad de disco magnético 151 y la unidad de disco óptico 155, por lo general, se conectan al bus de sistema 121 por medio de una interfaz de memoria extraíble, tal como la interfaz 150.

Las unidades y sus medios de almacenamiento informático asociados, que se han descrito en lo que antecede y que se ilustran en la figura 1, proporcionan el almacenamiento de instrucciones legibles por ordenador, estructuras de datos, módulos de programa y otros datos para el dispositivo informático 100. Por ejemplo, en la figura 1, la unidad de disco duro 141 se ilustra como que almacena el sistema operativo 144, los programas de aplicación 145, otros módulos de programa 146 y los datos de programa 147. Obsérvese que todos estos componentes pueden ser, o bien directamente o bien el mismo que o bien diferentes del sistema operativo 134, los programas de aplicación 135, otros módulos de programa 136 y los datos de programa 137. Al sistema operativo 144, los programas de aplicación 145, otros módulos de programa 146 y los datos de programa 147 se les dan números diferentes en el presente caso para ilustrar que, como mínimo, son copias diferentes.

Un usuario puede introducir órdenes e información en el dispositivo informático 100 a través de unos dispositivos de entrada, tales como un teclado 162 y un dispositivo apuntador 161, al que se hace referencia comúnmente como

ratón, bola de seguimiento o almohadilla táctil. Otros dispositivos de entrada (que no se muestran) pueden incluir un micrófono, una palanca de control, un controlador para juegos, una antena parabólica, un escáner o similares. Estos y otros dispositivos de entrada con frecuencia se conectan a la unidad de procesamiento 120 a través de una interfaz de entrada del usuario 160 que está conectada al bus de sistema 121, pero puede estar conectada por otra interfaz y estructuras del bus, tal como un puerto paralelo, un puerto de juegos, o un bus serie universal (USB, *universal serial bus*). Un monitor 191 u otro tipo de dispositivo de pantalla también está conectado al bus de sistema 121 por medio de una interfaz, tal como una interfaz de vídeo 190. Además del monitor 191 los ordenadores también pueden incluir otros dispositivos periféricos de salida, tales como unos altavoces 197 y una impresora 196, que pueden estar conectados a través de una interfaz de periféricos de salida 195.

Debido a que la tecnología de interconexión puede mejorar con el paso del tiempo, algunos dispositivos informáticos pueden contener unas interfaces heredadas para prever una compatibilidad con versiones anteriores de dispositivos heredados. El dispositivo informático 100 de la figura 1 se muestra con una interfaz heredada 198, que puede ser cualquiera de un número de interfaces, incluyendo un puerto serie, un puerto paralelo, un puerto de módem o similares. La interfaz heredada 198 puede habilitar que el dispositivo informático 100 se comunique con dispositivos heredado, tales como el dispositivo heredado 199, que puede ser una impresora, un escáner, un osciloscopio, un generador de funciones, o cualquier otro tipo de dispositivo de entrada o de salida. Tal como será conocido por los expertos en la materia, la mayor parte de los dispositivos de entrada o de salida modernos interaccionan a través de interfaces que dependen de unas normas de reciente desarrollo, tales como un puerto de USB o un puerto de IEEE 1394. No obstante, no es probable que los dispositivos heredados tengan tales interfaces y, por lo tanto, han de depender de una interfaz heredada con el fin de comunicarse con el dispositivo informático 100.

El dispositivo informático 100 puede operar en un entorno en red usando unas conexiones lógicas con uno o más ordenadores remotos. La figura 1 ilustra una conexión de red general 171 con un dispositivo informático remoto 180. La conexión de red general 171 puede ser cualquiera de diversos tipos diferentes de redes y conexiones de red, incluyendo una Red de Área Local (LAN, *Local Area Network*), una Red de Área Extensa (WAN, *Wide Area Network*), una red inalámbrica, redes que son conformes al protocolo de Ethernet, el protocolo de Anillo con Paso de Testigo, u otras redes lógicas, físicas o inalámbricas, incluyendo Internet o la World Wide Web.

Cuando se usa en un entorno de interconexión de redes, el dispositivo informático 100 está conectado con la conexión de red general 171 a través de un adaptador o interfaz de red 170, que puede ser una tarjeta de interfaz de red cableada o inalámbrica, un módem, o un dispositivo de interconexión de redes similar. En un entorno en red, los módulos de programa que se ilustran en relación con el dispositivo informático 100, o porciones del mismo, se pueden almacenar en el dispositivo de almacenamiento en memoria remoto. Se apreciará que las conexiones de red que se muestran son a modo de ejemplo y que se pueden usar otros medios de establecimiento de un enlace de comunicaciones entre los ordenadores.

En la descripción que se da en lo sucesivo, la invención se describirá con referencia a los actos y las representaciones simbólicas de las operaciones que son llevadas a cabo por uno o más dispositivos informáticos, a menos que se indique lo contrario. En ese sentido, se entenderá que tales actos y las operaciones, a los que a veces se hace referencia como que se ejecutan por ordenador, incluyen la manipulación por medio de la unidad de procesamiento del dispositivo informático de unas señales eléctricas que representan unos datos en una forma estructurada. Esta manipulación transforma los datos o los mantiene en unas ubicaciones en el sistema de memoria del dispositivo informático, el cual reconfigura o altera de otro modo el funcionamiento del dispositivo informático de una forma que es entendida por los expertos en la materia. Las estructuras de datos en las que se mantienen los datos son unas ubicaciones físicas de la memoria que tienen unas propiedades particulares que se definen por medio del formato de los datos. No obstante, a pesar de que la invención se está describiendo en el contexto anterior, no se tiene por objeto que la misma sea limitando, debido a que los expertos en la materia apreciarán que varios de los actos y las operaciones que se describen en lo sucesivo en el presente documento también se pueden poner en práctica en hardware.

Pasando a la figura 2, se ilustra un mecanismo que se contempla por medio de una forma de realización de la presente invención para aislar una extensión con respecto a una aplicación de software de host. Tal como se muestra en la figura 2, un proceso de host 201 puede invocar una entidad representante 205 en lugar de la propia extensión 215. La extensión 215 se puede alojar en un proceso virtual 211 que es distinto del proceso de host 201. El proceso virtual 211 puede intentar emular el proceso de host 201, al menos en la medida en la que el mismo pueda proporcionar las API de soporte virtuales 213 que sean análogas a las API de soporte 203 que pueden proporcionar la aplicación de software de host. La extensión 215, que se está ejecutando en el proceso virtual 211 puede, por lo tanto, usar las API de soporte virtuales 213 de la misma forma en la que esta usaría las API de soporte originales 203.

Un diseño para la entidad representante 205 que se contempla por medio de una forma de realización de la presente invención puede ser para emular la extensión 215, al menos en la medida en la que la entidad representante 205 pueda proporcionar unas API de servicio que sean análogas a las API de servicio que son proporcionadas por la extensión 215. El proceso de host 201 puede usar entonces las API que son proporcionadas por la entidad representante 205 para acceder a la funcionalidad de la extensión de la misma forma en la que este usaría las API de servicio que son proporcionadas por la propia extensión 215. No obstante, tal como se muestra en la figura 2,

cuando la entidad representante 205 recibe una solicitud a partir del proceso de host 201, usando una API de servicio de ese tipo, la entidad representante 205 puede recopilar la información relevante a partir del host y reenviar esa información a la extensión 215 que se está ejecutando dentro del proceso virtual 211.

Otro diseño para la entidad representante 205 que se contempla por medio de una forma de realización de la presente invención puede ser para interaccionar con el proceso de host 201 y traducir, o interceptar, determinadas funciones del proceso de host y utilizar la extensión 215 para extender la funcionalidad del proceso de host 201. Por ejemplo, la extensión 215 puede proporcionar acceso a un tipo particular de almacenamiento de archivos, tal como un almacenamiento de archivos usando un formato de sistema de archivos poco habitual o heredado. En un caso de ese tipo, una entidad representante 205 se puede diseñar para detectar unas instrucciones de acceso a archivos dentro del proceso de host 201 e interceptar esas instrucciones. La entidad representante 205 puede reenviar entonces la información apropiada a la extensión 215, que pueden acceder a los archivos en el almacenamiento de archivos usando el formato de sistema de archivos heredado. La información se puede devolver entonces a la entidad representante 205, desde la extensión 215, y la entidad representante 205 puede presentar la información al proceso de host 201. De tal forma, la entidad representante 205 puede extender la funcionalidad del proceso de host 201, tal como al habilitar que el proceso de host 201 acceda a los datos que se guardan en un formato de sistema de archivos heredado, incluso si el proceso de host no se diseñó para habilitar tal funcionalidad extendida. Por lo tanto, no es necesario que la entidad representante 205 esté basada en una extensión preexistente que se diseñó para interaccionar con el proceso de host 201, sino que más bien se puede diseñar para actuar como una corrección de compatibilidad entre el proceso de host y cualquier extensión.

Ya se diseñe la entidad representante 205 para emular una extensión preexistente, o para actuar como una corrección de compatibilidad para cualquier extensión, la entidad representante 205 puede reenviar la información apropiada a la extensión 215 con el fin de que la extensión ponga en práctica un trabajo para el proceso de host 201. Un procedimiento de reenvío de información desde la entidad representante 205 a la extensión 215 que se contempla por medio de una forma de realización de la presente invención solicita que la entidad representante 205 se comunique directamente con la extensión 215. En un caso de ese tipo, la propia entidad representante 205 puede invocar la API de servicio apropiada de la extensión 215. Un procedimiento alternativo de reenvío de la solicitud que se contempla por medio de una forma de realización de la presente invención solicita que la entidad representante 205 se comunique con un código auxiliar 217 que se está ejecutando dentro del proceso virtual 211. El código auxiliar 217 puede invocar entonces la API de servicio apropiada de la extensión 215. Tal como será conocido por los expertos en la materia, algunas extensiones pueden no manejar de forma correcta las solicitudes que se reciben por medio de una comunicación entre procesos. Para evitar tales dificultades, un código auxiliar, tal como el código auxiliar 217, dentro del proceso virtual 211 se puede usar para proporcionar un mecanismo por medio del cual la extensión 215 puede recibir solicitudes a través de sus API de servicio por medio de una comunicación dentro de un mismo proceso, en lugar de una comunicación entre procesos.

Una vez que la extensión 215 ha recibido la solicitud a partir del proceso de host 201, la misma puede proceder a responder a la solicitud. Dependiendo de la naturaleza de la solicitud, la extensión 215 puede acceder a una o más funciones que normalmente serían proporcionadas por el proceso de host 201 a través de las API de soporte 203, pero ahora pueden ser proporcionadas por el proceso virtual 211 a través de las API de soporte virtuales 213. Tal como se explicará con más detalle en lo sucesivo, dependiendo de la naturaleza de la solicitud del host, la extensión 215 puede necesitar acceder a los recursos del sistema informático 100 directamente, o acceder a los dispositivos de hardware que están conectados con el sistema informático de una forma directa. En un caso de ese tipo, se pueden hacer provisiones para conceder a la extensión 215 el acceso a tales recursos al tiempo que se sigue aislando la extensión 215 con respecto al proceso de host 201.

Para lograr el aislamiento previsto, puede que no sea suficiente tener meramente dos procesos separados, tales como el proceso de host 201 y el proceso virtual 211. Por lo tanto, algunas formas de realización de la presente invención contemplan que la entidad representante 205 se puede diseñar de una forma tal que se evite que las respuestas incorrectas a partir de la extensión 215, o un comportamiento incorrecto en la parte de la extensión, afecten al proceso de host 201. Por ejemplo, en un mecanismo que se contempla por medio de una forma de realización de la presente invención, la entidad representante 205 se puede diseñar para ajustarse de forma rigurosa a las API de servicio presentadas por la extensión 215. Por lo tanto, si la extensión 215 intenta devolver datos al proceso de host 201 que no es de la forma o el tipo que el host está esperando, la entidad representante 205 puede identificar el problema potencial y no pasar esos datos al proceso de host.

En otro mecanismo que se contempla por medio de una forma de realización de la presente invención, la entidad representante 205 puede aplicar una inteligencia adicional a los datos que se están devolviendo para evitar la introducción de inestabilidad en el proceso de host 201. Por ejemplo, si la extensión 215 experimenta un error fatal y falla, la entidad representante 205 puede mantener un contador de tiempo de expiración, o un mecanismo similar, para detectar el fallo de la extensión y puede informar al proceso de host 201 del error, tal como al proporcionar una respuesta de error o dejar de otro modo que el proceso de host se degrade de forma correcta sin, por ejemplo, perder el producto del trabajo de un usuario. La entidad representante 205 también puede devolver todo control que el proceso de host 201 pueda haber dado a la extensión 215, para evitar que el fallo de la extensión impida la ejecución del proceso de host. Por ejemplo, la entidad representante 205 puede solicitar que un sistema operativo subyacente termine el proceso virtual 211 y devuelva el control al proceso de host 201. Como alternativa, la entidad

representante 205 puede usar un código dedicado que es parte del proceso virtual 211 para informar al proceso virtual que, en apariencia, ha tenido lugar un fallo, y solicitar que el proceso virtual termine y devuelva el control al proceso de host 201.

No obstante, si la extensión 215 completa toda tarea que se le haya solicitado a la misma de forma correcta, esta puede devolver cualquier resultado que pueda ser esperado por el proceso de host 201 de la forma que se especifica por medio de la API de servicio. Por lo tanto, por ejemplo, si el resultado es una indicación de que la solicitud tuvo éxito, y se va a pasar en una variable previamente definida de vuelta al programa que realiza la llamada, la extensión 215 puede pasar esta variable de vuelta al código auxiliar 217 o directamente a la entidad representante 205. Desde ahí, la variable se puede devolver al proceso de host que realizó en un primer momento la llamada por medio de la entidad representante 205. De tal forma, la entidad representante 205 se puede volver indistinguible de la extensión 215, al menos en todo lo que respecta al proceso de host 201. Resulta obvio que, tal como será conocido por los expertos en la materia, pueden que algunas extensiones no necesiten devolver resultado alguno, caso en el cual no es necesario que se ponga en práctica provisión alguna para aceptar un valor devuelto.

Tal como se muestra en la figura 2, la extensión 215 opera en el proceso virtual 211. En consecuencia, si una acción de la extensión 215 da lugar a inestabilidad, será probable que la inestabilidad esté contenida en el interior del proceso virtual 211. En un caso de ese tipo, el sistema operativo o algún otro código, tal como la entidad representante 205, puede detectar el error en el proceso virtual 211 y puede terminar el mismo, o intentar reiniciarlo. En cualquier caso, no será probable que la inestabilidad afecte al proceso de host 201 y, por lo tanto, no dará como resultado un fallo perjudicial para el usuario. Por lo tanto, los mecanismos que se han descrito en lo que antecede permiten que el proceso de host 201 continúe operando de forma correcta incluso si la extensión 215 que está siendo usada por el proceso de host falla o se vuelve por lo demás inestable.

Tal como se ha descrito con detalle en lo que antecede, la entidad representante 205 puede presentar las API de servicio al proceso de host 201 de la misma forma en la que lo haría la extensión 215 si la misma se estuviera ejecutando en el proceso de host. En un mecanismo que se contempla por medio de una forma de realización de la presente invención, la entidad representante 205 se puede crear sobre la base de las API de servicio previamente definidas que se ponen en práctica por medio de la extensión 215. Tal como será conocido por los expertos en la materia, las API de servicio a través de las cuales una extensión y una aplicación de software de host pueden interoperar se conocen en general por adelantado debido a que el autor de la aplicación de software y el autor de extensiones a menudo son unas entidades diferentes. Cuando se instala una extensión, esta se puede registrar a sí misma con la aplicación de software de host, o un depósito de información apropiado, tal como la base de datos de registro 221, e indicar cuáles de las API de servicio soporta la misma. Usando esta información, la aplicación de software de host, o el sistema operativo subyacente, puede localizar la extensión apropiada cuando la aplicación de software de host intenta usar una de las API de servicio. Esta información también se puede usar para crear la entidad representante 205, debido a que la misma indica el conjunto completo de las API de servicio que son soportadas por la extensión 215. La creación de la entidad representante 205 también puede cambiar las entradas en, por ejemplo, la base de datos de registro 221, de una forma que se va a describir adicionalmente con detalle en lo sucesivo.

Otro mecanismo que se contempla por medio de una forma de realización de la presente invención es la creación de una “súper entidad representante” que puede aceptar solicitudes sobre la base de la totalidad del conjunto de API de servicio previamente definidas. Una súper entidad representante de ese tipo se puede invocar entonces con independencia de qué API de servicio particular busca usar la aplicación de host. En un caso de ese tipo, todo registro que la extensión 215 pueda poner en práctica en el momento de la instalación puede incluir un registro con la súper entidad representante, o la arquitectura de soporte subyacente, de tal modo que la súper entidad representante puede invocar la extensión correcta 215 cuando una API de servicio particular sea usada por la aplicación de software de host.

Un mecanismo adicional que se contempla por medio de una forma de realización de la presente invención es que la entidad representante 205 se puede crear sobre la base de la funcionalidad extendida que busca proporcionar la entidad representante al proceso de host 201. Por lo tanto, la entidad representante 205 se puede crear para detectar, interceptar o interaccionar de otro modo con una o más funciones que son usadas por o dentro del proceso de host 201 de tal modo que la entidad representante puede proporcionar los beneficios de la funcionalidad de la extensión 215 al proceso de host. Usando el ejemplo que se ha descrito en lo que antecede, si la entidad representante 205 se diseña para permitir que el proceso de host 201 acceda a un sistema de archivos heredado a través de la extensión 215, la entidad representante se puede diseñar para detectar e interceptar un acceso a archivos y funciones similares que son usadas por el proceso de host. La entidad representante 205 se puede diseñar adicionalmente para reenviar una información relevante a partir de esas funciones de acceso a archivo a la extensión 215 de tal modo que la extensión puede interaccionar con el sistema de archivos heredado. De forma similar, la entidad representante 205 se puede diseñar para aceptar respuestas a partir de la extensión 215 y convertir las mismas en un formato que sería reconocido por el proceso de host 201 como una respuesta apropiada que está asociada con las funciones de acceso a archivos interceptadas del proceso de host.

En algunos casos, puede ser deseable modificar las API de soporte virtuales 213 para reflejar de forma más precisa las API de soporte 203. Por ejemplo, las API de soporte virtuales 213 pueden, si son consultadas acerca de un identificador del proceso, devolver el identificador del proceso virtual 211. Puede, no obstante, ser deseable que las API de soporte virtuales 213 devuelvan el identificador del proceso de host 201. En un caso de ese tipo, se puede usar una comunicación de "canal posterior" o de "canal lateral" para habilitar que las API de soporte virtuales 213 accedan a la información a partir del proceso de host 201.

Para asegurar que la entidad representante correcta se invoca para la extensión particular que se solicita, una base de datos de registro, o un depósito de información similar, se puede usar para enlazar la entidad representante 205 con la extensión 215. Tal como se ha descrito en lo que antecede, la base de datos de registro 221 o un depósito de información similar, puede ser consultada por el proceso de host 201, o el sistema operativo, para determinar los parámetros para invocar la extensión 215. No obstante, en lugar de identificar la propia extensión 215, en su lugar la base de datos de registro 221 puede apuntar a la entidad representante 205.

Una vez que el proceso de host 201 ha invocado a la entidad representante 205, la entidad representante 205 puede proceder a invocar o a coordinar de otro modo la invocación de la extensión 215 dentro del proceso virtual 211. Tal como se describirá con detalle en lo sucesivo, el proceso virtual 211 se puede ya encontrar operativo o el mismo se puede encontrar en diversos estados de preparación. Si el proceso virtual 211 no se encuentra ya operativo, la entidad representante 205 puede coordinar la compleción de cualquier etapa que pudiera resultar necesaria para que el proceso virtual 211 alcance un estado operativo. Una vez que el proceso virtual 211 se encuentra operativo, la entidad representante 205 puede dar instrucciones al proceso virtual 211 para invocar la extensión 215. Por ejemplo, la entidad representante 205 puede proporcionar un puntero a la ubicación de la extensión 215 y puede transmitir los mismos parámetros, o unos similares, usados por el proceso de host 201. Además, si se determinó que la extensión 215 usa una comunicación de canal posterior o de canal lateral, cualquier recurso adicional que sea usado por la extensión también se puede invocar dentro del proceso virtual 211.

Una vez que el proceso virtual 211 ha invocado a la extensión 215, y cualquier otro código que sea usado por la extensión, la entidad representante 205 puede coordinar la invocación de un código auxiliar 217, si es necesario. Como alternativa, la entidad representante 205 puede establecer unos enlaces de comunicación con la extensión 215 directamente. Si se va a usar un código auxiliar 217, la entidad representante 205 puede dotar al proceso virtual 211 de la ubicación del código auxiliar 217 y los parámetros que se van a usar en la invocación del código auxiliar. Una vez que se ha invocado al código auxiliar 217, el propio código auxiliar puede establecer unos enlaces de comunicación con la extensión 215, así como establecer unos enlaces de comunicación con la entidad representante 205. La comunicación entre la entidad representante 205 y el código auxiliar 217 o la extensión 215 puede usar cualquier tipo de protocolos de comunicación entre procesos o dentro de un mismo proceso, incluyendo, por ejemplo, los conocidos mecanismos de Llamada a Procedimiento Remoto (RPC, *Remote Procedure Call*). A pesar de que es probable que los protocolos de comunicación que se usan se decidan por adelantado, se puede poner en práctica un procedimiento de toma de contacto para asegurar que la entidad representante 205 y el código auxiliar 217 o la extensión 215 se puedan comunicar de forma apropiada.

Debido a que algunas extensiones pueden depender de un contexto de modo de usuario para llevar a cabo las funciones que son solicitadas de las mismas por medio del proceso de host, puede que sea necesario proporcionar mecanismos por medio de los cuales una extensión en un entorno virtual se puede dotar de un contexto de modo de usuario. En general, un contexto de modo de usuario se puede referir al estado global de los recursos de un proceso, incluyendo memoria, archivos, entradas de registro, y similares de tal modo que las referencias de recurso particulares dentro de un contexto de modo de usuario dado sean precisas, mientras que esas mismas referencias, cuando se pasan fuera del contexto de modo de usuario particular, se pueden referir a unas ubicaciones de memoria incorrectas, o son por lo demás imprecisas. Para las extensiones que pueden aceptar o devolver unas cantidades grandes de datos, a menudo es más eficiente enviar y recibir unas referencias de memoria suponiendo un contexto de modo de usuario común, de lo que es enviar y recibir los propios datos. Por lo tanto, puede que se requiera mantener un contexto de modo de usuario común entre el proceso virtual 211 y el proceso de host 201 si una extensión que usa tales esquemas de paso de datos ha de operar de forma correcta.

Pasando a la figura 3, el proceso de host 201 se muestra habiendo invocado, de la forma que se ha descrito con detalle en lo que antecede, dos extensiones que se están ejecutando en el interior de los procesos virtuales 211 y 311, en concreto la extensión 215 y la extensión 315, de forma respectiva. La entidad representante 205 puede ser una súper entidad representante, tal como se ha descrito con detalle en lo que antecede, y puede dirigir solicitudes desde el proceso de host 201 o bien a la extensión 215 o bien a la extensión 315. Como alternativa, una segunda entidad representante, que no se muestra en la figura 3, se puede usar de tal modo que cada una de las extensiones 215 y 315 puede tener una relación de uno a uno con una entidad representante dentro del proceso de host 201.

En la figura 3 también se muestra el sistema operativo 134, que comprende la memoria de proceso de host 301 y las memorias de proceso virtual 302 y 303, que se corresponden con el proceso de host 201, el proceso virtual 211 y el proceso virtual 311, de forma respectiva. A pesar de que los mecanismos que se ilustran en las figuras 3 y 4 pueden depender de un sistema operativo común que subyace al proceso de host 201 y los procesos virtuales 211 y 311, unos mecanismos adicionales, que se describirán con mayor detalle en lo sucesivo, también pueden proporcionar un modo de usuario común entre el proceso de host y los procesos virtuales, incluso si los procesos virtuales se están

ejecutando independientemente del sistema operativo 134 que subyace al proceso de host. Cuando el proceso de host 201 y los procesos virtuales 211 y 311 sí comparten un sistema operativo común 134, tal como se ilustra en la figura 3, el sistema operativo también puede comprender una colección de asignaciones de tabla de páginas 320 que asignan la memoria de proceso de host 301 y las memorias de proceso virtual 302 y 303 a unos segmentos de la RAM física 132. A pesar de que la figura 3 muestra los segmentos 321, 322 y 323 como que se corresponden con la memoria de proceso de host 301 y las memorias de proceso virtual 302 y 303, de forma respectiva, será entendido por los expertos en la materia que los segmentos 321, 322 y 323 son únicamente ilustrativos y es probable que los segmentos físicos de RAM estén dispersos, y que no sean contiguos de la forma que se ilustra.

Para mantener un contexto de modo de usuario común entre el proceso de host 201 y los procesos virtuales 211 y 311, el sistema operativo 134, u otro software de soporte, puede proporcionar acceso a parte o la totalidad de los recursos que comprenden el contexto de modo de usuario del proceso de host 201 a los procesos virtuales 211 y 311. A pesar de que la siguiente descripción se centra en los mecanismos para proporcionar un acceso común a los aspectos de recursos de memoria de un contexto de modo de usuario, los expertos en la materia reconocerán la aplicabilidad de estos mecanismos a otros recursos que pueden comprender un contexto de modo de usuario, incluyendo recursos de registro, recursos de archivo, y similares.

En un mecanismo para proporcionar un acceso común a los aspectos de recursos de memoria de un contexto de modo de usuario que se contempla por medio de una forma de realización de la presente invención, el sistema operativo 134, o un software de soporte similar, puede copiar la memoria de proceso de host 301 en las memorias de proceso virtual 302 y 303. Tal como se ilustra en la figura 3, la copia de la memoria de proceso de host 301 en las memorias de proceso virtual 302 y 303 puede comportar una copia física del segmento de RAM 321 en unos nuevos segmentos de RAM 322 y 323. Como alternativa, el gestor de E / S puede copiar la memoria de proceso de host 301 en un grupo no paginado residente de memoria de sistema y puede proporcionar al proceso virtual 211 o 311 un acceso a ese grupo no paginado.

Una vez que la extensión 215 o 315 ha completado su tarea, la memoria de proceso virtual 302 o 303 se puede fusionar de nuevo con la memoria de proceso de host 301. Por ejemplo, la entidad representante 205 puede llevar a cabo una función de diferencia, que puede ser una comparación byte a byte, o una comparación a un nivel más macro, entre la memoria de proceso virtual en las ubicaciones 322 y 323 y la memoria de proceso de host en la ubicación 321 para determinar cualquier diferencia. Esas diferencias se pueden verificar como correctas y, por lo demás, conformes al comportamiento esperado de las extensiones 215 o 315 y se pueden copiar entonces de vuelta a la memoria de proceso de host 301, o facilitarse de otro modo al proceso de host 201 a través de la entidad representante 205. Como alternativa, si el gestor de E / S solo hubiera copiado la memoria de proceso de host 301 en un grupo no paginado residente de memoria de sistema, el gestor de E / S puede copiar el grupo no paginado de vuelta a la memoria de proceso de host. En general, tales copias se realizarían de una forma por solicitud. Por lo tanto, en lugar de copiar la totalidad de la memoria de proceso de host 301, un mecanismo más eficiente que se contempla por medio de una forma de realización de la presente invención solicita que el sistema operativo 134, u otro software de soporte, copie solo aquellas memorias intermedias de la memoria de proceso de host 301 que sean necesitadas por la extensión 215 o 315 para llevar a cabo la tarea solicitada. Cuando se llevan a cabo por el gestor de E / S del sistema operativo 134, tales copias específicas de la memoria intermedia en el grupo no paginado de memoria de sistema se conocen como “E / S Almacenada en Memoria Intermedia” o “Procedimiento de E / S Almacenado en Memoria Intermedia”.

Pasando a la figura 4, se muestra un mecanismo alternativo para proporcionar un acceso común a los aspectos de recursos de memoria de un contexto de modo de usuario que se contempla por medio de una forma de realización de la presente invención. En concreto, tal como se muestra en la figura 4, en lugar de copiar parte o la totalidad del proceso de memoria de host 301, las asignaciones de tabla de páginas 320 que son mantenidas por el sistema operativo 134 se pueden modificar para dirigir la memoria de proceso virtual 302 y 303 hacia la ubicación física 321 en la RAM 132 en la que se almacenan los datos que representan la memoria de proceso de host 301. Debido a que se elimina la necesidad de copiar datos, el mecanismo que se ilustra en la figura 4 puede ser más eficiente que el mecanismo que se ilustra en la figura 3.

No obstante, si las extensiones 215 y 315 pueden afectar a los segmentos físicos 321 que comprenden la memoria de proceso de host 301, un error o inestabilidad en la parte de las extensiones puede dar como resultado errores o inestabilidad en el propio proceso de host 201. Por lo tanto, para reducir al mínimo esta posibilidad, las asignaciones de tabla de páginas se pueden modificar de una forma de tipo “solo lectura” de tal modo que los procesos virtuales 211 y 311 se pueden apuntar a la memoria física 321 para leer la misma pero no se les permitirá modificar la misma. Ningún error o inestabilidad en la parte de las extensiones que se están ejecutando en los procesos virtuales 211 y 311 puede, por lo tanto, introducir error o inestabilidad alguna en el proceso de host 201 debido a que no se permitiría que los procesos virtuales modificaran la memoria del proceso de host.

Tal como se ha indicado en lo que antecede, la modificación a las asignaciones de tabla de páginas 320 que se contempla por medio del mecanismo de la figura 4 se puede realizar de una forma por solicitud. No obstante, si solo existe un proceso virtual, las asignaciones de tabla de páginas 320 pueden continuar apuntando hacia el segmento físico 321 de la RAM 132 incluso para solicitudes que no requieren un contexto de modo de usuario. La modificación de las asignaciones de tabla de páginas que se han descrito en lo que antecede se conoce en general como “E / S

Ni Almacenada en Memoria Intermedia Ni Directa” o “Procedimiento de E / S Ninguno”.

Un mecanismo alternativo adicional para proporcionar un acceso común a los aspectos de memoria de un contexto de modo de usuario que se contempla por medio de una forma de realización de la presente invención puede ser un híbrido de las alternativas que se ilustran en la figura 3 y la figura 4. En concreto, los procesos virtuales 211 y 311 se pueden dotar de un acceso de solo lectura a la memoria física 321, tal como se ha descrito con detalle en lo que antecede. No obstante, si o bien la extensión 215 o bien la extensión 315 necesita escribir datos de vuelta a la memoria, se puede poner en práctica una “copia en escritura”. Tal como será conocido por los expertos en la materia, una copia en escritura puede copiar los datos que se están modificando en una nueva ubicación antes de escribir la modificación en los datos. Por lo tanto, si la extensión 215 o la extensión 315 necesitara escribir datos de vuelta a la memoria 321, parte o la totalidad de la memoria 321 se puede copiar en una nueva ubicación, tal como 322 o 323, tal como se muestra en la figura 3, y la extensión 215 o la extensión 315 puede modificar entonces los datos copiados en la memoria 322 o 323. De tal forma, ningún error o inestabilidad que se introdujera por medio de las extensiones que se están ejecutando en los procesos virtuales 211 y 311 afectaría el proceso de host 201 debido a que no se permitiría que los procesos virtuales modificaran la memoria del proceso de host.

La entidad representante 205 puede realizar un seguimiento de aquellos segmentos de memoria que pueden haber sido editados por la extensión 215 o la extensión 315 usando los mecanismos de copia en escritura que se han descrito en lo que antecede. Cuando se accede a esos segmentos de memoria, la entidad representante puede hacer referencia de forma apropiada a las ubicaciones 322 o 323, en lugar de la ubicación 321. Si los datos que se almacenan en las ubicaciones 322 o 323 son conformes al comportamiento esperado de las extensiones 215 o 315, la entidad representante 205 puede permitir que los datos se usen dentro del proceso de host 201, tal como al copiar los mismos en la memoria de proceso de host 301, o al pasar las ubicaciones 322 o 323 al proceso de host. El aislamiento que se ha descrito en lo que antecede se puede lograr, por lo tanto, al tiempo que se permite que la entidad representante 205 acceda a los datos modificados.

Tal como se ha explicado en lo que antecede, la inicialización de un proceso virtual que puede alojar una extensión, tal como el proceso virtual 211 de la figura 2, se puede coordinar por medio de la entidad representante 205 después de que la entidad representante haya sido invocada por el proceso de host 201 en lugar de la extensión 215. Un tipo de proceso virtual que se contempla por medio de una forma de realización de la presente invención es una copia del proceso de host 201 que se está ejecutando en el mismo sistema operativo 134 que el proceso de host. Un proceso virtual de ese tipo se puede crear mediante la ramificación del proceso de host y el uso del proceso clonado como un proceso virtual. Como alternativa, se podría dar instrucciones al sistema operativo para que iniciara de nuevo cualquier aplicación de software que se hubiera invocado en un primer momento para crear el proceso de host 201. Por lo tanto, por ejemplo, si el proceso de host 201 era un navegador web, el proceso virtual 211 se podría crear mediante el inicio de la aplicación de navegador web de nuevo para crear un proceso separado o mediante la ramificación del proceso de navegador web que se encuentra en ejecución en la actualidad.

Otro tipo de proceso virtual que se contempla por medio de una forma de realización de la presente invención se puede crear dentro del contexto de un entorno de máquina virtual. Una máquina virtual puede ofrecer una solución óptima en el caso de que la extensión 215 fuera un controlador de dispositivo u otra extensión que fuera usada por un sistema operativo. A pesar de que puede ser posible usar un sistema operativo para crear otra copia de sí mismo para actuar como un proceso virtual, tal como mediante una ramificación o una nueva ejecución, una solución más elegante puede ser iniciar una máquina virtual y arrancar un sistema operativo en el entorno de la máquina virtual para actuar como un proceso virtual para alojar una o más extensiones. Es probable que un mecanismo de ese tipo prevea un mejor aislamiento y puede permitir que un sistema operativo use unas extensiones que estén diseñadas para un sistema operativo diferente. Por ejemplo, un controlador heredado que puede no haberse actualizado para una versión más reciente de un sistema operativo se puede alojar dentro de una versión más antigua del sistema operativo que se está ejecutando dentro de un entorno de máquina virtual. De tal forma, las características y capacidades de la extensión se pueden seguir facilitando a un usuario de un sistema operativo más nuevo, al tiempo que se protege al sistema operativo más nuevo frente a toda inestabilidad a la que pueda dar lugar la extensión heredada. Mediante el uso de una máquina virtual, o al llevar a cabo la ramificación o la nueva ejecución que se ha descrito en lo que antecede, el proceso virtual 211 puede proporcionar unas API de soporte equivalentes como el proceso de host 201 sin la necesidad de dar cuenta de las funciones de soporte de una forma individual.

A diferencia de los procesos virtuales 211 y 311, que reciben soporte a partir de un sistema operativo subyacente 134, una máquina virtual, tal como será conocido por los expertos en la materia, en general no hace uso de un sistema operativo de esta forma. En su lugar, para evitar la penalización de desempeño de hacer que cada instrucción de máquina virtual se pase a través de un sistema operativo completo, una máquina virtual puede depender, en su lugar, solo en un hipervisor que puede proporcionar una funcionalidad limitada de sistema operativo y puede abstraer el hardware subyacente del dispositivo informático para cualquiera que sea el sistema operativo que se ejecute en el entorno de máquina virtual. Mediante el uso de un hipervisor de ese tipo, una máquina virtual puede operar de una forma mucho más eficiente. No obstante, como consecuencia del uso de un hipervisor, antes de que el proceso de máquina virtual se pueda ejecutar en un procesador de un dispositivo informático, el sistema operativo de ese dispositivo informático se puede retirar y los apuntalamientos de ese sistema operativo se pueden almacenar. Posteriormente, cuando el proceso de máquina virtual ha completado una tarea, este puede retirar del hardware sus apuntalamientos, y el sistema operativo original se puede restablecer. Un intercambio de ese tipo del

uso de hardware, entre el sistema operativo de un dispositivo informático, y un proceso de máquina virtual, puede tener lugar muchas veces por segundo. Por lo tanto, a pesar de que el usuario puede percibir la máquina virtual como simplemente otra aplicación que usa el sistema operativo, el proceso de máquina virtual en general solo comparte en el tiempo el hardware de dispositivo informático con el sistema operativo.

Para lograr el intercambio que se ha descrito en lo que antecede, una máquina virtual puede comprender un controlador de dispositivo de máquina virtual o una extensión similar que pueda ser invocada por el sistema operativo del dispositivo informático. El controlador de dispositivo de máquina virtual puede proporcionar las instrucciones necesarias para retirar los apuntalamientos del sistema operativo con respecto al hardware de dispositivo informático y almacenar los mismos en memoria caché hasta un instante tal que se permita que el sistema operativo reanude la ejecución. Además, el controlador de dispositivo de máquina virtual puede coordinar la invocación del proceso de máquina virtual. Por ejemplo, el sistema operativo puede recibir, mientras que el mismo se está ejecutando, una orden de usuario para hacer que el proceso de máquina virtual lleve a cabo una tarea. El sistema operativo puede emitir entonces una orden al controlador de dispositivo de máquina virtual para hacer que el proceso de máquina virtual lleve a cabo la tarea solicitada y devuelva el control al sistema operativo de una forma eficiente. Por lo tanto, el sistema operativo puede tratar el paso del control al proceso de máquina virtual de la misma forma en la que este trataría el paso del control a cualquier otro subproceso que actualmente se esté coordinando por medio del sistema operativo. El controlador de dispositivo de máquina virtual puede, tras la recepción de una orden de ese tipo, retirar los apuntalamientos del sistema operativo con respecto al hardware de dispositivo informático, permitir que el hipervisor instale sus apuntalamientos, y pasar la orden al proceso de máquina virtual. Posteriormente, cuando el proceso de máquina virtual se ha completado, el controlador de dispositivo de máquina virtual puede volver a instalar los apuntalamientos del sistema operativo y permitir que el mismo reanude la ejecución en el hardware de dispositivo informático.

Tal como se ha descrito con detalle en lo que antecede, la entidad representante 205 puede detectar un fallo dentro del proceso virtual 211, y puede buscar evitar que ese fallo introduzca inestabilidad en el proceso de host 201. No obstante, si el proceso virtual 211 es un proceso de sistema operativo virtual que se está ejecutando en un entorno que es creado por una máquina virtual, puede ser difícil que la entidad representante 205 detecte o controle un proceso de sistema operativo virtual de ese tipo, debido a que el sistema operativo del que puede depender la entidad representante 205 no se está ejecutando en el hardware de dispositivo informático, sino que en su lugar se almacena y está aguardando a que la máquina virtual complete su ejecución. En consecuencia, un mecanismo de aislamiento de errores que se contempla por medio de una forma de realización de la presente invención solicita que el hipervisor supervise el software que se está ejecutando en el entorno que es creado por la máquina virtual y que detecte los fallos dentro de ese entorno. Si se detecta un fallo, el hipervisor puede detener la ejecución, volver a instalar los apuntalamientos del sistema operativo, y permitir que el mismo reanude la ejecución en el hardware de dispositivo informático. El hipervisor también puede proporcionar una respuesta apropiada para permitir que el sistema operativo, u otro software que estaba dependiendo de la extensión en el entorno virtual, se degrade de forma correcta.

Además, debido a que el sistema operativo en general no puede reanudar la ejecución hasta que al mismo se le permita hacerlo por parte del hipervisor, el hipervisor también puede mantener un temporizador o un mecanismo similar para asegurar que un fallo en el entorno de máquina virtual no impida que el control vuelva jamás al sistema operativo. A pesar de que un mecanismo de temporizador se puede usar para detectar un fallo, de la forma que se ha descrito en lo que antecede, el mecanismo de temporizador puede tener una importancia adicional si una máquina virtual se usa para crear un entorno en el que alojar una o más extensiones debido a que puede que no exista ningún otro mecanismo por medio del cual se puede devolver el control al sistema operativo si un fallo tiene lugar en el entorno de máquina virtual.

Como alternativa, en lugar de mantener un mecanismo por medio del cual se pueden detectar los fallos, tal como un mecanismo de temporizador, en el hipervisor, un mecanismo de ese tipo se puede mantener en el hardware del dispositivo informático 100, lo que puede solicitar al hipervisor que devuelva el control al sistema operativo si se detecta un fallo en el entorno que es creado por la máquina virtual. Por ejemplo, el sistema operativo puede establecer un temporizador en hardware antes de permitir que el hipervisor se ejecute en el hardware. Posteriormente, si un fallo tiene lugar dentro del entorno que es creado por la máquina virtual, el temporizador mantenido por hardware puede expirar y solicitar al hipervisor que devuelva el control al sistema operativo. Para devolver el control al sistema operativo, el hipervisor se puede modificar para abortar toda ejecución si el temporizador mantenido por hardware expira, y devolver el control al sistema operativo. El hipervisor también puede indicar la presencia de un error, o puede indicar que una ejecución no se completó si el control se devuelve de esta forma.

Una complicación adicional, si el proceso virtual 211 es un proceso de sistema operativo virtual que se está ejecutando en un entorno que es creado por una máquina virtual, es que la comunicación entre la entidad representante 205 y el proceso virtual 211, o la extensión 215, puede no ser capaz de depender de los mecanismos de comunicación entre procesos o de RPC, tal como se ha descrito con detalle en lo que antecede. En su lugar, la comunicación entre la entidad representante 205 y el proceso de sistema operativo virtual 211 se puede coordinar por medio del hipervisor u otros mecanismos que se configuran por medio de la máquina virtual para comunicarse con el proceso de sistema operativo que subyace al proceso de host 201. Tales mecanismos pueden incluir, por

ejemplo, almacenar mensajes en unas ubicaciones de memoria previamente definidas con el fin de que puedan acceder a las mismas tanto la máquina virtual como el sistema operativo cuando cada uno se está ejecutando en el hardware de dispositivo informático o, como otro ejemplo, proporcionar subprocesos de comunicación que permanecen en memoria al tiempo que tanto la máquina virtual como el sistema operativo se están ejecutando en el hardware de dispositivo informático.

Además, los mecanismos que se han descrito con detalle en lo que antecede, que pueden proporcionar un modo de usuario común entre el proceso virtual 211 o 311 y el proceso de host 201, también pueden requerir que se ponga en práctica alguna modificación en un entorno en el que el proceso virtual 211 o 311 es un proceso de sistema operativo virtual que se está ejecutando dentro de un entorno de máquina virtual. Por ejemplo, en lugar de depender de un sistema operativo común 134 para llevar a cabo las modificaciones a las asignaciones de tabla de páginas, las modificaciones se pueden hacer en las asignaciones de tabla de páginas que son mantenidas por el hipervisor de la máquina virtual. Por lo tanto, si la memoria de proceso de host 301 se copia para crear la memoria de proceso virtual 302 y 303, una copia de ese tipo puede ser llevada a cabo por el hipervisor en lugar del sistema operativo 134 que se muestra en la figura 3. Más en concreto, la memoria de proceso de host 301 puede permanecer en la ubicación de memoria física 321 incluso después de que ya no se esté ejecutando el sistema operativo de host y se esté ejecutando el proceso de máquina virtual. El hipervisor puede identificar la ubicación de memoria física 321, y puede copiar los contenidos de esa ubicación en una ubicación de memoria física 322 o 323 que se puede encontrar bajo el control del hipervisor.

De una forma similar, si el modo de usuario común entre el proceso de host 201 y los procesos virtuales 211 y 311 se logra mediante la modificación de las asignaciones de tabla de páginas, de la forma que se ha descrito con detalle en lo que antecede con referencia a la figura 4, la modificación de las asignaciones de tabla de páginas puede ser llevada a cabo por el hipervisor. Por lo tanto, la memoria de proceso de host 301 puede permanecer en la ubicación de memoria física 321 y el hipervisor puede asignar la memoria de proceso virtual 302 y 303 a la ubicación de memoria física 321 incluso si el sistema operativo de host no se está ejecutando actualmente. Resulta significativo que ambas de las memorias de proceso virtual que sería necesario que se asignaran a la ubicación física 321, tales como la memoria de proceso virtual 302 o 303, se encontrarían bajo el control del hipervisor. En consecuencia, debido a que la memoria de proceso de host 301 no requeriría modificación alguna, el mecanismo anteriormente descrito no requiere soporte alguno a partir del sistema operativo 134, que puede ser, por lo tanto, cualquier sistema operativo convencional.

Si la memoria de proceso virtual se asigna a las ubicaciones de memoria física que son usadas por la memoria de proceso de host y un esquema de copia en escritura, tal como el que se ha descrito con detalle en lo que antecede, se va a usar, el hipervisor también puede llevar a cabo la copia necesaria. Por ejemplo, el hipervisor puede dejar al margen una ubicación de memoria física adicional en la que almacenar unos valores que se escriben como parte de la copia en escritura. Además, tal como se ha descrito en lo que antecede, la entidad representante 205 se puede modificar para hacer referencia tanto a la memoria de proceso de host 301 como a las ubicaciones adicionales que se usan para la copia en escritura. No obstante, debido a que la memoria adicional que es dejada al margen por el hipervisor puede no ser una memoria que pueda ser usada por el sistema operativo que subyace a la entidad representante 205, la entidad representante se puede modificar para hacer referencia, de forma específica, a las ubicaciones de memoria incluso si no se accede a las mismas de forma correcta por medio del sistema operativo subyacente. Como alternativa, las ubicaciones de memoria que son dejadas al margen por el hipervisor se pueden copiar adicionalmente en unas ubicaciones de memoria a las que puede acceder el sistema operativo que subyace a la entidad representante 205 como parte del procedimiento por medio del cual la máquina virtual deja de ejecutarse en el dispositivo informático y se permite al sistema operativo que reanude la ejecución.

Un mecanismo alternativo adicional para proporcionar un contexto de modo de usuario común que se contempla por medio de una forma de realización de la presente invención solicita que un proceso de host suplente se ejecute en el interior del proceso de sistema operativo virtual. Por ejemplo, un proceso de host suplente, que es análogo al proceso de host, se puede ejecutar encima del sistema operativo virtual en el entorno de máquina virtual. El contexto de modo de usuario del proceso de host suplente puede ser idéntico al contexto de modo de usuario del proceso de host que se encuentra en el exterior del entorno de máquina virtual, previendo de forma automática, de ese modo, un modo de usuario común. El modo de usuario común puede ser mantenido por la comunicación entre el proceso de host y el proceso de host suplente, tal como mediante el uso de las técnicas que se han descrito en lo que antecede, sin la necesidad de acceder a o copiar de forma explícita la memoria de proceso de host 301.

Un mecanismo que se contempla por medio de una forma de realización de la presente invención para crear un proceso de sistema operativo virtual, es la invocación de una aplicación de software de máquina virtual en el dispositivo informático de host 100, seguida por el arranque de un sistema operativo apropiado dentro del contexto del entorno que se crea cuando se ejecuta la aplicación de software de máquina virtual. Tal como será conocido por los expertos en la materia, una aplicación de software de máquina virtual en general comprende una extensión de sistema operativo que se puede usar para retirar los apuntalamientos del sistema operativo 134 con respecto al hardware de dispositivo informático y almacenar los mismos en un almacenamiento temporal. Una aplicación de software de máquina virtual también puede comprender un hipervisor que, después de que se hayan retirado los apuntalamientos del sistema operativo 134, puede instalar sus propios apuntalamientos en el hardware de dispositivo informático y abstraer ese hardware de una forma apropiada para crear un entorno virtual. Un sistema

operativo virtual, que puede ser el mismo o diferente del sistema operativo 134, se puede arrancar entonces en el hardware abstraído que es proporcionado por el hipervisor. Por lo tanto, el hipervisor puede crear un entorno de máquina virtual en el que un proceso de sistema operativo virtual se puede ejecutar independientemente del sistema operativo 134. A pesar de que un proceso de sistema operativo virtual de ese tipo puede proporcionar los beneficios que se han enumerado en lo que antecede, la invocación de una aplicación de software de máquina virtual, incluyendo la retirada descrita del sistema operativo 134, y el arranque de un sistema operativo apropiado dentro del entorno de máquina virtual, puede ser un proceso prohibitivamente lento.

Para evitar la falta de eficiencia que se introduce mediante el inicio de una aplicación de software de máquina virtual y, a continuación, arrancar un sistema operativo dentro del entorno de máquina virtual, otro mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que se inicialice una máquina virtual y que se arranque un sistema operativo dentro del entorno de máquina virtual y que el estado final resultante del entorno de máquina virtual se guarde y se clone para un uso adicional. Por lo tanto, por ejemplo, durante un inicio inicial del dispositivo informático 100, después de que se haya arrancado el sistema operativo 134, una aplicación de software de máquina virtual se puede iniciar de forma automática y un sistema operativo virtual se puede arrancar dentro del entorno que es creado por la máquina virtual. Una vez que se ha arrancado este sistema operativo virtual, el estado del entorno de máquina virtual se puede guardar. Tal como será conocido por los expertos en la materia, un estado de ese tipo se puede guardar con facilidad debido a que es probable que la aplicación de software de máquina virtual solo cree un puñado de archivos en los medios de almacenamiento del dispositivo informático 100 que comprenden el estado del entorno de máquina virtual. Se puede acceder a esos archivos y los mismos se pueden copiar y la aplicación de software de máquina virtual se puede dejar entonces en un estado operativo o, como alternativa, la misma se puede colocar en un estado de reserva, tal como un modo de inactividad, o esta incluso se puede apagar totalmente.

Posteriormente, cuando un proceso de host, que puede ser el sistema operativo 134 o cualquiera de las aplicaciones de software 145, intenta poner en práctica una operación que daría como resultado el uso de una extensión, o bien por diseño, o bien debido a que puede haber intercedido una entidad representante, se puede copiar el estado guardado del entorno de máquina virtual y se puede crear de una forma eficiente un entorno de máquina virtual nuevo. Debido a que el estado del entorno de la máquina virtual ya comprende un sistema operativo virtual arrancado, un proceso virtual que puede alojar la extensión solicitada se puede crear con facilidad. Por ejemplo, si la extensión solicitada es una extensión de sistema operativo, un proceso virtual para la extensión, ya existe en la forma del sistema operativo virtual. Si, por otro lado, la extensión solicitada es una extensión de aplicación de software, entonces la aplicación de software apropiada se puede ejecutar en el sistema operativo virtual y puede crear, por lo tanto, un proceso virtual apropiado. En consecuencia, al guardar el estado que se crea por medio de una aplicación de software de máquina virtual después de que un sistema operativo virtual haya sido arrancado dentro del entorno de la máquina virtual y, a continuación, clonar ese estado guardado según sea necesario, un proceso virtual para alojar unas extensiones tanto de aplicación de software como de sistema operativo se puede crear de forma eficiente.

Para proporcionar un soporte apropiado para la creación de un proceso virtual, la aplicación de software de máquina virtual se puede diseñar para abstraer un superconjunto de hardware que puede ser más grande que lo que normalmente abstraería una aplicación de software de máquina virtual de ese tipo. De forma similar, el sistema operativo virtual que se arranca dentro del entorno de máquina virtual puede poner en práctica un conjunto de API de sistema operativo completo. Mediante la abstracción de un superconjunto de hardware de ese tipo, y la provisión de un conjunto de API de sistema operativo completo, hay una mayor probabilidad de que el estado que se crea por medio de la máquina virtual se pueda usar para generar un proceso virtual apropiado para una extensión solicitada. En consecuencia, un número mayor de procesos virtuales útiles se puede generar mediante la clonación del estado guardado, y será necesario crear menos procesos virtuales usando unos mecanismos más costosos.

Pasando a la figura 5, se muestra otro mecanismo para crear un proceso de sistema operativo virtual que se contempla por medio de una forma de realización de la presente invención. El diagrama de flujo 400 en general ilustra los procedimientos de inicio de muchos dispositivos informáticos modernos, tales como el dispositivo informático 100. No se tiene por objeto que el diagrama de flujo 400 sea una descripción detallada del proceso de inicio de un sistema operativo o dispositivo informático particular, sino que, en su lugar, se tiene por objeto que proporcione una ilustración general de los elementos que comúnmente se hallan en los procedimientos de inicio, con el fin de explicar mejor los mecanismos que se contemplan por medio de una forma de realización de la presente invención.

Tal como se puede ver a partir de la figura 5, se inicia un procedimiento de inicio al proporcionar alimentación al dispositivo informático en la etapa 405. En una etapa 410 posterior, una Unidad de Procesamiento Central (CPU, *Central Processing Unit*) puede comenzar a ejecutar las instrucciones que se hallan en el Sistema Básico de Entrada / Salida (BIOS, *Basic Input / Output System*) de la Memoria de Solo Lectura (ROM, *Read Only Memory*). La BIOS de ROM puede llevar a cabo unas pruebas básicas de hardware para asegurar que los elementos de hardware centrales de un dispositivo informático están funcionando de forma correcta. En la etapa 415, la BIOS puede leer la información de configuración, que se almacena en general en una memoria de Metal - Óxido - Semiconductor Complementario (CMOS, *Complementary Metal - Oxide Semiconductor*). Tal como será conocido por los expertos en la materia, la memoria de CMOS puede ser un área pequeña de memoria cuyos contenidos son

mantenidos por una batería cuando el dispositivo informático no se encuentra operativo. La memoria de CMOS puede identificar uno o más medios legibles por ordenador que se pueden conectar con el dispositivo informático. Tal como se indica por medio de la etapa 420, la BIOS puede examinar el primer sector de diversos medios legibles por ordenador en un esfuerzo por hallar un Registro Maestro de Arranque (MBR, *Master Boot Record*).

En general, el MBR contiene parte o la totalidad de un cargador de particiones, que pueden ser unas instrucciones ejecutables por ordenador para localizar un registro de arranque y comenzar el arranque de un sistema operativo. Por lo tanto, en la etapa 425 el cargador de particiones que se halló en el MBR puede hacerse con el control de la BIOS y puede examinar una tabla de particiones, o un registro similar, en el medio legible por ordenador, para determinar un sistema operativo apropiado que cargar. Cada sistema operativo puede tener un registro de arranque asociado con el mismo y, en la etapa 430, si el registro de arranque no tiene problema alguno, el cargador de particiones puede iniciar el arranque del sistema operativo.

Como parte del arranque del sistema operativo, el cargador de particiones puede invocar unas rutinas de detección de hardware que pueden comenzar a poner en práctica la detección de hardware, tal como se indica por medio de la etapa 435. En general, la detección de hardware que se lleva a cabo en la etapa 435 solo es preliminar y, en lugar de habilitar necesariamente el hardware, la detección de hardware de la etapa 435 solo puede crear una lista de dispositivos de hardware para un uso posterior. Una lista de ese tipo se puede almacenar, por ejemplo, en una base de datos de registro o un depósito de información similar. En la etapa 440, el cargador de particiones puede invocar otro subsistema o proceso de sistema operativo para proporcionar un enlace de comunicación y de control a los diversos dispositivos de hardware del dispositivo informático. A veces, este subsistema se conoce como “Capa de Abstracción de Hardware” (HAL, *Hardware Abstraction Layer*). Además, el cargador de particiones también puede, en la etapa 440, cargar el kernel del sistema operativo y el registro, o una base de datos similar que contiene la información necesaria de hardware y de software.

El registro, o una base de datos similar que se cargue por medio del cargador de particiones en la etapa 440, también puede contener una lista de controladores de dispositivo que pueden ser necesarios para que el kernel de sistema operativo acceda al hardware requerido, tal como la unidad de disco duro o la memoria. En la etapa 445, por lo tanto, el cargador de particiones puede cargar estos controladores de dispositivo con el fin de proporcionar el soporte apropiado para el kernel de sistema operativo. Una vez que se han cargado los controladores de dispositivo, el cargador de particiones puede, también en la etapa 445, transferir el control del dispositivo informático al kernel de sistema operativo.

A pesar de que las etapas 405 a 445 del diagrama de flujo 400 han ilustrado, en general, los elementos de la mayor parte de las rutinas de arranque, la etapa 450 ilustra la primera parte de un mecanismo que se contempla por medio de una forma de realización de la presente invención para crear un proceso de sistema operativo virtual que puede alojar extensiones de sistema operativo o aplicaciones de software. En concreto, en la etapa 450, la HAL o una información que está asociada con el registro de arranque puede indicar, al kernel de sistema operativo, que en el dispositivo informático se encuentran presentes más CPU de las que se encuentran, de hecho, físicamente presentes. Por lo tanto, por ejemplo, en un dispositivo informático con solo una única CPU, el kernel de sistema operativo puede recibir, en la etapa 445, una indicación de dos o más CPU que se encuentran presentes en el dispositivo informático. De forma similar, para un dispositivo informático que ya tiene dos CPU, el kernel de sistema operativo puede recibir una indicación de tres o más CPU que se encuentran presentes en el dispositivo informático. Tal como se describirá con detalle en lo sucesivo, al indicar la presencia de unas CPU que no se encuentran, de hecho, presentes, un proceso de sistema operativo virtual se puede crear con más facilidad y eficiencia.

Volviendo al diagrama de flujo 400, en la etapa 455 el kernel de sistema operativo puede llamar a la HAL para inicializar cada CPU que el kernel de sistema operativo cree que se encuentra presente en el dispositivo informático. La solicitud para inicializar una CPU puede incluir, por lo tanto, unas CPU que no se encuentran, de hecho, presentes en el dispositivo informático. Una vez que la HAL ha completado la inicialización de la totalidad de las CPU, el estado del sistema se puede guardar, en la etapa 460, para su uso posterior en la creación, de forma eficiente, de un proceso de sistema operativo virtual, de una forma que se va a describir con detalle en lo sucesivo. El arranque del sistema operativo puede continuar entonces con unas operaciones de inicio convencionales, incluyendo, por ejemplo, inicializar diversos subsistemas del sistema operativo, activar los dispositivos de hardware que comprenden el dispositivo informático 100, y cargar los controladores de dispositivo apropiados, tal como se indica por medio de la etapa 465. A pesar de que la etapa 465 enumera de forma específica la inicialización de un subsistema de entrada / salida (E / S), el kernel de sistema operativo también puede inicializar gestores de memoria, gestores de proceso, gestores de objetos, diversos kernel del sistema operativo, y subsistemas similares en la etapa 465. Además, el kernel de sistema operativo puede volver a habilitar las interrupciones de hardware y puede activar los diversos dispositivos de hardware que se detectan como parte del dispositivo informático 100. Tal como se ha indicado en lo que antecede, como parte de la activación de diversos dispositivos de hardware, el kernel de sistema operativo también puede cargar los controladores de dispositivo apropiados para esos dispositivos. Tal como será conocido por los expertos en la materia, debido a que muchos sistemas operativos se diseñaron en un primer momento para un dispositivo informático con una única CPU, tales sistemas operativos llevan a cabo en general la mayor parte de las etapas que se ilustran en la figura 5 con solo una única CPU, y solo activar cualquier CPU adicional después de casi completar la totalidad de los procedimientos de inicio. En consecuencia, la CPU primaria mantiene en general la totalidad de los enlaces de hardware, mientras que a las otras CPU se pueden poner al

cargo de diversos procesos que se estarán ejecutando en el dispositivo informático.

Tal como se ha descrito en lo que antecede, en la etapa 450, se informó al kernel de sistema operativo de las CPU adicionales incluso a pesar de que las CPU pueden no haberse encontrado físicamente presentes en el dispositivo informático. Por lo tanto, en la etapa 470, se puede informar al kernel de sistema operativo de que han fallado aquellas CPU que se indicaron en la etapa 450, pero no se encuentran físicamente presentes. Esta indicación de las CPU con errores en la etapa 470 deshace, en efecto, la indicación de las CPU adicionales en la etapa 450, y permite que el kernel de sistema operativo complete el proceso de arranque del sistema operativo usando el mismo número de CPU que se encuentran físicamente presentes en el dispositivo informático 100. Tal como se ha indicado en lo que antecede, debido a que diversos sistemas pueden inicializar las CPU adicionales en una diversidad de instantes, no se tiene por objeto que la etapa 470 se limite a tener lugar después de que se haya llevado a cabo la totalidad de los elementos que se ilustran en la etapa 465. En su lugar, se tiene por objeto que la etapa 470 se lleve a cabo después de que se hayan inicializado las CPU adicionales y se hayan establecido los enlaces de hardware apropiados, siempre que eso pueda tener lugar. Procediendo con el diagrama de flujo 400, en la etapa 475, el kernel de sistema operativo puede iniciar un subsistema apropiado para crear el entorno de modos de usuario y en la etapa 480, una vez que se ha creado el entorno de modos de usuario, el sistema operativo puede completar el proceso de arranque.

Una vez que se ha completado el proceso de arranque en la etapa 480, un entorno virtual se puede arrancar, tal como mediante la ejecución de una máquina virtual por medio de unas órdenes que se introducen a través del sistema operativo cuyo arranque se completó en la etapa 480. Para crear el entorno virtual de forma más eficiente, se puede usar el estado que se guardó en la etapa 460 durante el arranque del sistema operativo. Debido a que el estado guardado refleja las múltiples CPU que se presentan en la etapa 450, y no tiene en cuenta la indicación de los fallos de las CPU secundarias en la etapa 470, el entorno virtual se puede arrancar como si las múltiples CPU se encontraran presentes. Por lo tanto, el entorno de la máquina virtual se puede aprovechar, de la forma que se muestra en lo sucesivo, de los mecanismos que se establecen por medio del sistema operativo de host para iniciarse de forma más eficiente.

Debido a que, tal como se ha indicado en lo que antecede, muchos sistemas operativos usarán solo una única CPU hasta que el proceso de arranque casi se ha completado, esa CPU se pone en general al cargo de manejar la mayor parte o la totalidad de los dispositivos de sistema, incluyendo manejar toda comunicación, tal como las interrupciones de hardware, a partir de esos dispositivos de sistema. En consecuencia, un sistema operativo en un dispositivo informático que tiene múltiples CPU físicas en general proporciona mecanismos por medio de los cuales los procesos que se están ejecutando en una CPU que no se usa durante el proceso de arranque se pueden comunicar con la CPU que se usa durante el proceso de arranque, con el fin de proporcionar a esos procesos la capacidad de comunicarse con hardware. La figura 5 ilustra un mecanismo que puede aprovechar esta capacidad para permitir que el entorno de una máquina virtual se comunique con el hardware subyacente sin tener enlace en tiempo de ejecución alguno con los dispositivos de hardware. En concreto, cuando el estado guardado se proporciona al entorno virtual, el entorno virtual se puede configurar de tal modo que no se usa la CPU que se hubiera usado durante el proceso de arranque o, al menos, no se le permite comunicarse con un hardware de entrada / salida. En su lugar, el entorno virtual puede usar los mecanismos del sistema operativo para aprovechar los enlaces de hardware que ya se han llevado a cabo para el sistema operativo al comportarse como si el dispositivo informático comprendiera múltiples CPU.

Como un ejemplo, en un dispositivo informático que solo tiene una única CPU, el proceso de sistema operativo virtual operará como si hubiera al menos una segunda CPU debido a que, mientras que el sistema operativo habría recibido una indicación, en la etapa 470, de que la segunda CPU ha fallado, el entorno virtual no habría recibido indicación alguna de ese tipo. Por lo tanto, a pesar de que la única CPU física en el dispositivo informático sigue llevando a cabo la totalidad del trabajo, el entorno de la máquina virtual opera como si existiera un sistema de dos CPU, con una CPU que tiene la totalidad de los enlaces en tiempo de ejecución con los dispositivos de hardware, y una segunda CPU que aloja el proceso de sistema operativo virtual, que, debido a la existencia de la primera CPU, no es necesario que se inicialice con enlace en tiempo de ejecución alguno con el hardware. Como resultado, el sistema operativo virtual se puede arrancar de forma eficiente debido a que el mismo no necesita inicializar hardware alguno y la propia máquina virtual se puede iniciar de una forma muy eficiente debido a que la misma no necesita abstraer hardware alguno. Si una extensión que está alojada dentro del proceso de sistema operativo virtual requiere una comunicación con un dispositivo de hardware, se puede hacer una solicitud desde el proceso de sistema operativo virtual al sistema operativo de host usando los mecanismos anteriormente descritos que se establecen para su uso en los sistemas de múltiples CPU. Por lo tanto, la extensión puede operar de una forma convencional, y el entorno virtual se puede crear de forma eficiente.

No obstante, tal como será conocido por los expertos en la materia, para algunas extensiones, tales como controladores de dispositivo de sistema operativo, el mecanismo que se ha descrito en lo que antecede puede no proporcionar una solución satisfactoria. En concreto, si el sistema operativo de host encuentra un hardware heredado, tal como el dispositivo heredado 199, puede que este no sea capaz de localizar un controlador apropiado y puede no reconocer el hardware de forma correcta. Por lo tanto, mientras que un proceso de sistema operativo virtual apropiado puede alojar un controlador de dispositivo heredado, tal como la interfaz heredada 198, puede que no haya forma alguna de comunicarse con el hardware heredado debido a que, usando los mecanismos

anteriormente descritos, el sistema operativo manejaría la totalidad de la comunicación de hardware, y el sistema operativo no se hubiera conectado de forma correcta con el hardware heredado. Además, incluso si el sistema operativo subyacente se hubiera conectado de forma correcta con la totalidad del hardware del dispositivo informático, algunas extensiones, tales como controladores de dispositivo de vídeo, pueden no ser capaz de operar de forma correcta con incluso la cantidad mínima de retardo que se introduce en las comunicaciones de hardware usando los mecanismos anteriores.

En consecuencia, una variante del mecanismo anteriormente descrito que se contempla por medio de una forma de realización de la presente invención solicita que el dispositivo de hardware cuyo controlador de dispositivo se alojará en un proceso de sistema operativo virtual se identifique durante la secuencia de arranque del sistema operativo subyacente y se enlace, no con el sistema operativo subyacente, sino con el proceso de sistema operativo virtual, proporcionando al controlador de dispositivo un acceso directo a ese dispositivo de hardware. Más en concreto, las interrupciones el dispositivo de hardware se pueden enviar a una CPU secundaria que se indica, pero no se encuentra físicamente presente. Posteriormente, cuando una máquina virtual crea un entorno suponiendo que la CPU secundaria sí existe, este será capaz de inicializar un enlace en tiempo de ejecución con el dispositivo de hardware, permitiendo que el proceso de sistema operativo virtual se comunique directamente con el dispositivo de hardware. Por lo tanto, tal como se muestra en la figura 5, antes de la compleción del arranque del entorno virtual en la etapa 499, una etapa opcional 495 puede insertar la configuración de hardware del dispositivo heredado 199 y puede cargar el controlador de dispositivo correcto, tal como la interfaz heredada 198, en el entorno virtual.

Como alternativa, la máquina virtual puede crear un entorno con dos o más CPU virtuales sin depender de la optimización de arranque que se ha descrito en lo que antecede. Con independencia del proceso que se usa para crear el entorno virtual de múltiples CPU, un dispositivo de hardware cuyo controlador de dispositivo es alojado por un proceso de sistema operativo virtual se puede enlazar como si el dispositivo de hardware estuviera enviando interrupciones a una CPU secundaria que es una CPU virtual. Por lo tanto, durante el arranque inicial del sistema operativo, el dispositivo de hardware cuyo controlador se debería alojar en un entorno virtual se puede ocultar o retardar, tal como se describirá con detalle adicional en lo sucesivo, de tal modo que el dispositivo de hardware no se enlaza con la CPU física que está cargando el sistema operativo. El entorno virtual, no obstante, como parte del proceso de arranque, se puede enlazar con el dispositivo de hardware. Tal como se ha explicado en lo que antecede, el entorno virtual se puede crear como si existiera al menos una segunda CPU y el entorno virtual la estuviera usando. Por lo tanto, el enlace con el dispositivo de hardware se llevará a cabo como si el dispositivo de hardware estuviera enviando interrupciones a la segunda CPU. Debido a que solo existe una única CPU física, esta puede recibir comunicaciones a partir del dispositivo de hardware. No obstante, esas comunicaciones se pueden dirigir hacia el entorno virtual en lugar del sistema operativo de host, dotando al entorno virtual de un acceso directo al dispositivo de hardware.

Algunas formas de realización de la presente invención contemplan un número de mecanismos por medio de los cuales el dispositivo de hardware cuyo controlador se debería alojar en un proceso de sistema operativo virtual se puede ocultar o retardar en la etapa 465 del diagrama de flujo 400. Un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita la captura de toda información de control que se pueda enviar, durante la etapa 465, al controlador de dispositivo que se debería alojar en un proceso de sistema operativo virtual. Tal información de control se puede retardar hasta que el proceso de sistema operativo virtual se ha establecido en la etapa 490 y, a continuación, retransmitirse al controlador de dispositivo. Otro mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que la entidad representante del controlador de dispositivo, que sería invocada por el proceso de sistema operativo de la forma que se ha descrito en lo que antecede con referencia al proceso de host 201 y la entidad representante 205, devuelva una indicación de “CORRECTO” en la etapa 465 y, posteriormente, almacene en memoria caché todo Paquete de Solicitud de Entrada / salida (IRP, *Input / output Request Packet*) que se envía a la misma hasta que el proceso de sistema operativo virtual se hubo establecido en la etapa 490. La entidad representante podría reenviar entonces los IRP al controlador de dispositivo en el proceso de sistema operativo virtual. Como alternativa, la entidad representante podría simplemente demorarse hasta que se hubiera establecido el proceso de sistema operativo virtual, y podría pasar entonces todo IRP directamente al controlador de dispositivo sin requerir un almacenamiento en memoria caché.

Aún otro mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que el dispositivo de hardware se enlace en un primer momento con el sistema operativo en la etapa 465 y, posteriormente, se le envíe una orden de “hibernar”, o una similar, que pueda vaciar limpiamente todo IRP en la cola y dejar el hardware en un estado conveniente. El controlador de dispositivo en el proceso de sistema operativo virtual puede entonces, en la etapa 495, intentar establecer una comunicación directa con el dispositivo desde dentro del proceso de sistema operativo virtual. Una variante de este mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que el dispositivo de hardware se oculte al sistema operativo en la etapa 465, en lugar de enlazarse y, a continuación, hibernarse, tal como se ha descrito en lo que antecede. Un dispositivo de hardware se puede ocultar mediante el envío de unas órdenes apropiadas a la HAL, o diversos otros subsistemas, tales como un gestor de tipo conectar y usar. Posteriormente, después de que el sistema operativo se haya arrancado en la etapa 480 y el proceso de sistema operativo virtual se haya establecido, el dispositivo de hardware se puede activar, o hacerse de otro modo visible en la etapa 495 y, por lo tanto, se puede enlazar a sí mismo con el proceso de sistema operativo virtual y al controlador de dispositivo que se aloja en el mismo.

En lugar de intentar simular unas CPU adicionales para aprovechar las capacidades de los sistemas operativos de múltiples CPU de la forma que se ha descrito con detalle en lo que antecede, un mecanismo alternativo para crear de forma eficiente un proceso virtual que se contempla por medio de una forma de realización de la presente invención se ilustra, en general, en la figura 6. El diagrama de flujo 500 que se ilustra en la figura 6 contiene muchas de las mismas etapas que se han descrito con detalle en lo que antecede con referencia a la figura 5. En concreto, las etapas 405 a 445 y 465 y 475 ilustran, en general, los mismos procedimientos básicos de inicio que se han descrito con detalle en lo que antecede. Además, a pesar de que no se ilustra de forma específica en la figura 6, el kernel de sistema operativo puede, entre las etapas 445 y 465, aprender acerca de las CPU del dispositivo informático, y puede llamar a la HAL para inicializar esas CPU. No obstante, a diferencia de las etapas 450 y 455 que se ilustran en la figura 5, las etapas que se han descrito en lo que antecede no comportan la presentación de un número mayor de CPU al kernel de sistema operativo que las que, de hecho, existen en el dispositivo informático. Posteriormente a la etapa 475, se puede poner en práctica una nueva etapa 505 por medio de la cual se puede guardar el estado del dispositivo informático.

Después de que el arranque de sistema operativo se haya completado en la etapa 485, se puede iniciar una máquina virtual, y la máquina virtual puede aprovechar la información que se recopila por medio del código de observación y de registro. Por lo tanto, en la etapa 485, la máquina virtual puede comenzar el proceso de arranque y, en la etapa 510, la máquina virtual puede usar el estado que se registra en la etapa 505 para arrancar de forma más eficiente un proceso de sistema operativo virtual. Más en concreto, el entorno virtual puede usar los parámetros de solo los dispositivos de hardware particulares que este necesita virtualizar, lo que permite que el mismo omita otros dispositivos de hardware. Además, debido a que los parámetros ya se han establecido y registrado durante el arranque de sistema operativo, tal como en la etapa 505, la máquina virtual puede virtualizar esos dispositivos de hardware de forma más eficiente. Si, no obstante, un dispositivo de hardware, tal como el dispositivo heredado 199, no se inicializó de forma correcta en la etapa 465, este se puede inicializar en el entorno virtual en una etapa opcional 495, de la forma que se ha descrito con detalle en lo que antecede. En última instancia, debido a que la máquina virtual puede seleccionar un conjunto limitado de dispositivos de hardware que virtualizar, y puede virtualizar los mismos de forma más eficiente, un entorno virtual se puede crear de forma más eficiente. No obstante, tal como será reconocido por los expertos en la materia, la optimización que se ha descrito en lo que antecede puede ser de lo más eficaz si el sistema operativo arrancado y el sistema operativo virtual son idénticos, o al menos similares, en sus interfaces con el hardware.

En algunos casos, incluyendo determinadas extensiones de controlador de dispositivo de hardware que puedan ser alojadas por un proceso de sistema operativo virtual, la semántica de las API de soporte que son proporcionadas por el proceso de sistema operativo virtual puede no ser útil. Por ejemplo, algunos controladores de dispositivo de hardware pueden requerir un acceso al hardware físico con el fin de controlar el mismo de forma correcta. Por lo tanto, en estos casos será necesario que el proceso de sistema operativo virtual proporcione a los controladores de dispositivo alojados un acceso al hardware físico. A pesar de que algunos de los mecanismos que se han descrito en lo que antecede pueden proporcionar el acceso directo necesario, algunas formas de realización de la presente invención contemplan unos mecanismos adicionales que se pueden aplicar a cualquier proceso virtual para permitir que las extensiones que están alojadas dentro de ese proceso tengan un acceso directo al hardware.

En consecuencia, los mecanismos que se describen con detalle en lo sucesivo se pueden usar, no solo para proporcionar un aislamiento frente a fallos entre una extensión y un proceso de host, sino también para posibilitar que las máquinas virtuales proporcionen un acceso directo al hardware en situaciones en las que la abstracción del hardware puede ser poco eficiente o imposible. Por ejemplo, los mecanismos anteriores pueden permitir que una máquina virtual aloje un software que depende de hardware para abstraer el cual no se ha diseñado la máquina virtual. En ese sentido, los mecanismos anteriores proporcionan a los diseñadores y autores de las máquinas virtuales la capacidad de estrechar la gama del hardware del que los mismos necesitan dar cuenta al tiempo que se sigue proporcionando a los consumidores la capacidad de usar un hardware único o heredado.

Pasando a la figura 7, se muestra un proceso de máquina virtual 617, usando un hipervisor 613 para interaccionar con el hardware subyacente 620, y comprendiendo un proceso de sistema operativo virtual 611 que aloja una extensión 615. Tal como se indica por medio de la flecha de color negro, algunas formas de realización de la presente invención contemplan un entorno de máquina virtual de tal modo que la extensión 615 puede acceder directamente al hardware 620 desde dentro del entorno de máquina virtual, sorteando toda abstracción que sea llevada a cabo por el hipervisor 613. Tal como se ha explicado en lo que antecede, un hipervisor, tal como el hipervisor 613, pueden ser las instrucciones ejecutables por ordenador que gestionan un entorno de máquina virtual al proporcionar una funcionalidad limitada de sistema operativo y al proporcionar un acceso abstraído al hardware subyacente, tal como el hardware 620. Por lo tanto, el hipervisor 613 puede actuar para proteger al entorno de máquina virtual frente a los detalles específicos del hardware subyacente, permitiendo que la aplicación de software de máquina virtual cree un entorno de máquina virtual apropiado para cualquier código cuya ejecución se tenga por objeto dentro del mismo. El hipervisor puede traducir entonces entre el entorno de máquina virtual y el hardware subyacente.

Como un ejemplo, el entorno de máquina virtual puede presentar un tipo particular de CPU al proceso de sistema operativo virtual 611, y cualquier programa que se pudiera ejecutar dentro de ese proceso, mientras que el hardware subyacente 620 podría, de hecho, comprender un tipo completamente diferente de CPU. El hipervisor 613 se puede

poner al cargo de traducir las solicitudes que se realizan a un tipo de CPU en el interior del entorno de máquina virtual a las solicitudes apropiadas para comunicarse con los diferentes tipos de CPU que se encuentra presente en el hardware subyacente 620. No obstante, tal como se ha explicado en lo que antecede, debido a que algunas extensiones de sistema operativo, tales como los controladores de dispositivo, pueden necesitar comunicarse directamente con los dispositivos de hardware subyacente, la abstracción que es llevada a cabo por el hipervisor puede evitar que tales extensiones de sistema operativo operen de forma correcta. En consecuencia, algunas formas de realización de la presente invención contemplan diversos mecanismos para sortear el hipervisor y permitir que las extensiones que están alojadas dentro del proceso de sistema operativo virtual 611 accedan directamente al hardware.

Además del proceso de máquina virtual 617, la figura 7 también ilustra un proceso de sistema operativo de host 601 que también puede usar el hardware 620. El hardware 620 se separa en dos bloques para ilustrar la compartición en el tiempo que se ha descrito en lo que antecede entre el proceso de sistema operativo de host 610 y el proceso de máquina virtual 617. Por lo tanto, mientras el proceso de máquina virtual 617 se está ejecutando, por medio del hipervisor 613, en el hardware 620, el hardware 620 tampoco está ejecutando de forma simultánea el proceso de sistema operativo de host 601. En su lugar, los apuntalamientos del proceso de sistema operativo de host 601 se pueden haber retirado y colocado en un almacenamiento temporal. A pesar de que no se ilustran en la figura 7, tales apuntalamientos pueden incluir entradas de registro, diversos registros de control, rutinas de envío de interrupción, datos de privilegios de CPU, y similares. Una vez que el proceso de máquina virtual 617 ha acabado de ejecutarse en el hardware 620, los apuntalamientos del proceso de máquina virtual se pueden retirar y colocarse en un almacenamiento temporal y el proceso de sistema operativo de host 601 se puede restablecer y se le puede permitir que se ejecute en el hardware.

A pesar de que la figura 7 ilustra, de hecho, el proceso de sistema operativo de host 601, con la entidad representante 605, los mecanismos para proporcionar un acceso directo al hardware desde dentro de un entorno virtual que se contempla por medio de las formas de realización de la presente invención se pueden usar en el exterior del contexto del aislamiento frente a fallos de extensión. En concreto, los mecanismos anteriores se pueden aplicar a la tecnología de máquinas virtuales en general, permitiendo que las máquinas virtuales alojen extensiones y otro software que depende de dispositivos de hardware heredados, dispositivos de hardware personalizados o dispositivos de hardware atípicos. Al eliminar la necesidad de diseñar una abstracción para tales dispositivos, algunas formas de realización de la presente invención prevén unos hipervisores más simples, y unos diseños de máquina virtual más eficientes.

Un mecanismo para proporcionar un acceso directo al hardware desde dentro de un entorno de máquina virtual que se contempla por medio de una forma de realización de la presente invención solicita que el hipervisor modifique la asignación de tabla de páginas para permitir un acceso a la memoria física que se corresponde con uno o más dispositivos de hardware. Tal como será conocido por los expertos en la materia, una aplicación o extensión se puede comunicar con los dispositivos de hardware mediante el acceso a una memoria física apropiada, que a menudo pueden ser los registros o un hardware similar que está ubicado o bien en el propio dispositivo de hardware o en una tarjeta de interfaz. Por lo tanto, por ejemplo, el dispositivo informático 100 ilustrativo que se muestra en la figura 1 puede permitir que un controlador de dispositivo de teclado se comunique con el teclado 162 al proporcionar al controlador de dispositivo de teclado un acceso a los registros de memoria física de la interfaz de entrada de usuario 160. Como alternativa, el controlador de dispositivo de teclado puede acceder a una ubicación particular en la RAM 132 y procesos adicionales pueden transferir la entrada a partir del teclado 162 a esa ubicación en la RAM con el fin de ser leída por el controlador de dispositivo.

Cuando un código en un entorno de máquina virtual, tal como la extensión 615 en el proceso de máquina virtual 617, busca acceder al hardware subyacente, el hipervisor 613 puede llevar a cabo unas traducciones apropiadas para el hardware subyacente y o bien puede acceder a los registros físicos por sí mismo o bien puede almacenar los datos en el espacio de memoria de proceso de máquina virtual, a partir del cual los mismos se pueden leer y copiar en los registros físicos apropiados por medio de un hardware dedicado o similares. Para proporcionar un acceso directo a los dispositivos de hardware subyacente desde dentro de un entorno de máquina virtual, el hipervisor puede evitar la puesta en práctica de toda traducción, debido a que tales traducciones pueden ser incorrectas y, en su lugar, el hipervisor puede modificar las asignaciones de tabla de páginas de una forma tal que las ubicaciones de memoria física necesarias se pueden asignar en el espacio de memoria apropiado, tal como el espacio de memoria que es usado por el proceso de sistema operativo virtual 611. Tal como se ha explicado con detalle en lo que antecede, las asignaciones de tabla de páginas determinan qué ubicaciones de memoria física se asignan a unos procesos dados. Por lo tanto, mediante la modificación de las asignaciones de tabla de páginas para colocar, en el espacio de memoria de proceso de sistema operativo virtual, las ubicaciones de memoria física que se corresponden con uno o más dispositivos, el hipervisor puede permitir que las extensiones y aplicaciones que usan el sistema operativo virtual accedan de forma directa a los dispositivos de hardware.

En un ejemplo, una extensión 615, que puede ser un controlador de dispositivo de hardware, y está siendo alojada por un proceso de sistema operativo virtual 611, puede obtener un acceso directo a un dispositivo de hardware correspondiente, que es parte del hardware 620, usando unas operaciones de lectura y de escritura en memoria conocidas. El hipervisor 613, que proporciona las abstracciones de hardware, se puede diseñar para reconocer las operaciones de lectura y de escritura en memoria a partir de la extensión 615 como unas operaciones que no se

deberían traducir o abstraer de otro modo, y les puede permitir pasar a su través hasta el hardware subyacente. Además, debido a que el hipervisor 613 puede modificar las asignaciones de tabla de páginas, según sea apropiado, las operaciones de lectura y de escritura en memoria se pueden poner en práctica físicamente en los registros previstos o en otras ubicaciones de memoria física que se corresponden con el dispositivo de hardware que busca controlar la extensión 615. En consecuencia, la extensión 615 tiene un control directo sobre los registros de memoria u otras ubicaciones de memoria física que se corresponden con el dispositivo de hardware y, por lo tanto, puede controlar directamente el dispositivo incluso desde dentro del entorno de máquina virtual.

No obstante, al cambiar las asignaciones de tabla de páginas, y permitir que las extensiones accedan directamente al hardware desde dentro de un entorno de máquina virtual, el proceso de sistema operativo de host 601 puede quedar más expuesto a toda inestabilidad que se pueda introducir por medio de la extensión. Por ejemplo, mientras el proceso de máquina virtual 617 se está ejecutando en el hardware 620, la extensión 615 puede acceder directamente un cierto componente del hardware 620 de una forma incorrecta, dando lugar a que ese componente de hardware se comportara de forma no apropiada, o incluso que se volviera inoperable. Posteriormente, después de que el proceso de sistema operativo de host 601 haya reanudado la ejecución en el hardware 620, el componente de hardware al que se ha accedido se puede continuar comportando de forma no apropiada, y posiblemente, introduciendo inestabilidad en el proceso de sistema operativo de host, o el mismo puede permanecer inoperable y, por lo tanto, evitar que el proceso de sistema operativo de host ponga en práctica una tarea requerida. En consecuencia, un mecanismo que se contempla por medio de una forma de realización de la presente invención prevé ciertas limitaciones en las modificaciones de asignación de tabla de páginas que se han descrito en lo que antecede. Por ejemplo, una limitación puede ser modificar la asignación de tabla de páginas solo en la medida que necesite la extensión. Por lo tanto, si una extensión solo requiere un acceso a un intervalo de direcciones muy limitado, que comprende posiblemente las direcciones de los registros de memoria que están ubicados físicamente en el dispositivo de hardware, o en una interfaz con el dispositivo, entonces las asignaciones de tabla de páginas se pueden modificar solo en la medida que sea necesaria para asignar ese intervalo de direcciones limitado al espacio de memoria de proceso de máquina virtual. Otra limitación puede ser una limitación temporal, por medio de la cual las asignaciones de tabla de páginas se pueden modificar solo mientras se permita que la extensión lleve a cabo su tarea. Por ejemplo, cuando la extensión 615 se intenta comunicar directamente con los dispositivos de hardware, la misma puede realizar una solicitud del hipervisor 613 indicando el periodo de tiempo durante el cual esta desea un acceso directo. Una solicitud de ese tipo se puede realizar directamente o a través del proceso de sistema operativo virtual 611 que aloja la extensión 615. Una vez que el hipervisor 613 ha recibido la solicitud, este puede modificar las asignaciones de tabla de páginas durante el periodo de tiempo solicitado.

Tal como será conocido por los expertos en la materia, muchos dispositivos de hardware están conectados con un dispositivo informático a través de un hardware de interfaz, tal como tarjetas de interfaz y similares. A menudo, tal hardware de interfaz está acoplado con unos mecanismos de bus conocidos, tales como los que se han descrito en lo que antecede. Las direcciones de bus se pueden asignar a una memoria física a la que se puede acceder adicionalmente por medio de un software que se está ejecutando en el dispositivo informático. En consecuencia, a menudo se hace referencia a los registros de las tarjetas de interfaz, y similares, que están conectados con el bus como “registros asignados a la memoria”, y se pueden asignar a una o más páginas físicas de memoria. No obstante, debido a que un conjunto de registros asignados a la memoria rara vez comparte una página física con otro conjunto de registros asignados a la memoria, las modificaciones anteriores a las asignaciones de tabla de páginas se puede realizar de una forma por dispositivo.

Además, un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita el uso de una traducción de dirección virtual para permitir que determinados registros asignados a la memoria sean facilitados solo al proceso de máquina virtual 617. De tal forma, el proceso de sistema operativo de host 601 puede evitar abordar un hardware para el cual puede que este no tenga un controlador de dispositivo correcto, y al controlador de dispositivo correcto, que se puede alojar dentro de un proceso de sistema operativo virtual se le puede conceder un acceso permanente al dispositivo de hardware particular.

Otro mecanismo para proporcionar a las máquinas virtuales un acceso directo al hardware que se contempla por medio de una forma de realización de la presente invención permite que se acceda a los puertos de entrada / salida (E / S) desde dentro del entorno de máquina virtual sin emulación alguna u otras modificaciones que sean llevadas a cabo por el hipervisor 613. Tal como será conocido por los expertos en la materia, en general los puertos de E / S se identifican por medio de una dirección, o número de puerto, y se puede acceder a los mismos por medio de unas órdenes conocidas de “ENTRADA” o de “SALIDA”. Para que los controladores de dispositivo u otras aplicaciones de software accedan a los dispositivos de hardware usando puertos de E / S, las órdenes de ENTRADA y de SALIDA o bien se pueden reenviar, a través de software, a los registros o puertos físicos en el dispositivo de hardware que se especificaron en las órdenes o bien, como alternativa, las mismas se pueden pasar a los registros o puertos identificados directamente desde el controlador de dispositivo u otras aplicaciones que emiten las órdenes. Algunos tipos de CPU prevén un paso a través o acceso directo selectivo mediante el uso de un mapa de bits de E / S en el segmento de tarea, en el que el mapa de bits de E / S especifica unas direcciones para las cuales las instrucciones se pueden pasar a través de software y unas direcciones para las cuales las instrucciones se pueden enviar directamente a los registros o puertos físicos.

Durante el funcionamiento normal, el hipervisor de una máquina virtual, tal como el hipervisor 613, o bien realizará una captura sobre las instrucciones de E / S o bien emulará las instrucciones de E / S para abstraer de forma correcta el hardware subyacente 620 para un software dentro del entorno de máquina virtual. Si el hipervisor 613 realiza una captura sobre las instrucciones de E / S usando, por ejemplo, un mapa de bits de protección, un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que una modificación del mapa de bits de protección proporcione unos “agujeros”, o unas direcciones de E / S para las cuales el hipervisor no realizará una captura. Por lo tanto, por ejemplo, si la extensión 615, que puede ser un controlador de dispositivo, requiere un acceso directo al hardware usando una dirección de E / S particular, entonces el mapa de bits de protección puede detectar las instrucciones de E / S desde dentro del proceso de máquina virtual 617, tal como a partir de la extensión 615, que especifican esa dirección de E / S, y el mapa de bits de protección puede permitir que esas instrucciones de E / S pasen a través del hipervisor sin captura.

No obstante, si el hipervisor 613 emula las instrucciones de E / S, entonces un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita una modificación del hipervisor de tal modo que se puede realizar una comprobación antes de la emulación y, para las instrucciones de E / S que especifican unas direcciones particulares, no se llevará a cabo emulación alguna. Por lo tanto, si, por ejemplo, la extensión 615 requiere un acceso directo al hardware en una dirección de E / S particular, el hipervisor 613 puede comprobar las direcciones de E / S que se especifican en las instrucciones de E / S recibidas, y si las instrucciones de E / S recibidas especifican la dirección particular que es usada por la extensión, el hipervisor puede permitir que esas instrucciones de E / S pasen a su través sin emulación alguna. De tal forma, una extensión puede tener un acceso directo al hardware incluso desde dentro de un entorno de máquina virtual.

Tal como se puede ver, los mecanismos anteriormente descritos pueden proporcionar a las extensiones y otras aplicaciones de software un acceso directo al hardware a través de puertos de E / S incluso desde dentro de un entorno de máquina virtual. No obstante, si las extensiones u otras aplicaciones de software no se diseñan para acceder al hardware directamente a través de puertos de E / S y, en su lugar, dependen del sistema operativo para llevar a cabo tal acceso de hardware, un mecanismo que se contempla por medio de una forma de realización de la presente invención prevé una modificación del hipervisor 613 de tal modo que, cuando el proceso de sistema operativo virtual 611 detecta una solicitud a partir de la extensión 615, u otra aplicación de software que requeriría que el proceso de sistema operativo virtual accediese directamente al hardware 620 a través de un puerto de E / S, este puede pasar esa solicitud al hipervisor, que puede poner en práctica entonces la instrucción de E / S apropiada en nombre de la extensión u otra aplicación de software. Como alternativa, el proceso de sistema operativo virtual 611 puede llevar a cabo la propia instrucción de E / S y el hipervisor 613 puede dejar que la instrucción pase a su través, tal como mediante el uso de los mecanismos que se han descrito con detalle en lo que antecede.

Otro mecanismo que se usa a menudo para comunicarse con el hardware se conoce como Acceso Directo a Memoria (DMA, *Direct Memory Access*). Tal como será conocido por los expertos en la materia, un DMA puede permitir que un controlador de dispositivo, u otra aplicación de software, pase datos a o desde un dispositivo de hardware sin sobrecargar la CPU. Más en concreto, un DMA prevé la transferencia de datos desde uno o más segmentos de memoria física a los registros físicos, o elementos similares, del propio dispositivo de hardware. Una transferencia de ese tipo se coordina por medio de un conjunto de circuitos en el dispositivo informático, tal como chips de DMA dedicados, pero no requiere coordinación alguna por parte de la CPU.

En general, las solicitudes de DMA pueden ser parte de la API de soporte que se proporciona a una extensión por medio de un sistema operativo o una aplicación de software. No obstante, debido a que la API de soporte virtual que se ha descrito en lo que antecede puede ser proporcionada por un proceso de sistema operativo virtual que se está ejecutando dentro de un entorno de máquina virtual, las direcciones de memoria que se especifican por medio de un DMA que se origina en el interior del entorno de máquina virtual pueden no ser la dirección física correcta a la que se debería dirigir el dispositivo de hardware. Esto puede ser debido a un número de factores, en particular que la dirección de DMA se pueda haber modificado por medio del hipervisor como parte de la abstracción de hardware que es llevada a cabo por el hipervisor. En consecuencia, para que un DMA se ponga en práctica de forma correcta, las direcciones físicas correctas se pueden usar dentro del entorno de máquina virtual.

Un mecanismo para proporcionar la dirección física correcta para un DMA que se contempla por medio de una forma de realización de la presente invención solicita que el hipervisor 613 o el proceso de sistema operativo virtual 611 proporcione, a la extensión 615, unas regiones de memoria que son adecuadas para un acceso de DMA por medio del hardware. Además, para proteger frente a las solicitudes de DMA malintencionadas o incorrectas, el hipervisor 613 también puede bloquear o desviar de otro modo a unas direcciones correctas cualquier DMA que apunte a unas direcciones que deberían estar protegidas. Las direcciones protegidas se pueden determinar, por ejemplo, por adelantado tal como cuando el hipervisor 613 se ejecuta en primer lugar en el hardware 620. Las direcciones protegidas también pueden ser simplemente aquellas direcciones de memoria que pueden no ser capaces de proporcionar el soporte necesario para una comunicación de DMA con otros dispositivos de hardware. Como aún otra alternativa, las direcciones protegidas pueden ser una cualquiera o la totalidad de las direcciones que no están participando en la solicitud de DMA actual. A menudo, evitar el uso de las direcciones protegidas en un DMA se puede poner en práctica por medio de unos chips de DMA dedicados, bus de memoria, o un conjunto de circuitos similar, en el propio dispositivo informático 100. En un caso de ese tipo, el hipervisor 613 puede aprender acerca de estos bloques y usar los mismos, en lugar de intentar bloquear o desviar un DMA por medio de una solución de

software.

Con el fin de proporcionar unas direcciones de memoria que son adecuadas para un DMA a la extensión 615, un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que el hipervisor 613 supervise el funcionamiento de la extensión 615 y detecte los DMA próximos. Como alternativa, el proceso de sistema operativo virtual 611 puede supervisar el funcionamiento de la extensión y o bien proporcionar una información relevante al hipervisor 613, o bien el propio sistema operativo virtual puede detectar los DMA próximos. Tal como se ha explicado en lo que antecede, las extensiones usan, en general, unas API de soporte para obtener un acceso a diversos recursos. Por lo tanto, un DMA próximo se puede detectar mediante la supervisión de las funciones que son llamadas por la extensión 615 a través de las API de soporte virtuales que son proporcionadas por el proceso de sistema operativo virtual 611. Determinadas funciones conocidas se usan, en general, para configurar un DMA, tal como, por ejemplo, una solicitud para establecer un bloque de memoria o una solicitud para una dirección física de memoria, En consecuencia, se puede determinar que es probable que una extensión que solicita esas funciones a partir de una API de servicio virtual se esté preparando para llevar a cabo un DMA.

En lugar de supervisar de forma continua las llamadas a función de API de servicio virtual que son realizadas por la extensión 615, el hipervisor 613 o el proceso de sistema operativo virtual 611, puede detectar de forma más eficiente un DMA posible mediante la modificación de la API de soporte virtual para incluir una instrucción ilegal cuando se invocan las funciones conocidas que se usan, en general, para configurar un DMA. Una instrucción ilegal de ese tipo puede generar entonces una captura y alertar al hipervisor o el proceso de sistema operativo virtual acerca del DMA próximo.

Una vez que el hipervisor 613 o el proceso de sistema operativo virtual 611 ha tomado conciencia de un DMA próximo, tal como mediante el uso de los mecanismos anteriormente descritos, este puede proporcionar un intervalo apropiado de direcciones de memoria a la extensión 615, permitiendo que el DMA proceda de forma correcta. En algunos casos, el hipervisor 613 puede llevar a cabo un intercambio de memoria o una gestión de memoria similar con el fin de ser capaz de proporcionar un intervalo apropiado de direcciones de memoria. Como alternativa, el hipervisor 613 puede depender de unas capacidades conocidas de dispersión / recopilación del dispositivo informático de host para colocar, en un intervalo de memoria apropiado, la información que se va a enviar a, o que se va a recibir a partir de, el dispositivo de hardware por medio de un DMA. No obstante, debido a que la extensión 615 espera unas direcciones poco habituales debido a la traducción que, en general, es llevada a cabo por el hipervisor 613, resulta poco probable que las maquinaciones adicionales que se han descrito en lo que antecede afecten de forma adversa a la extensión.

Una vez que las direcciones de memoria se han proporcionado a la extensión 615, puede ser necesario evitar que procesos adicionales accedan a la memoria en esas direcciones hasta que el DMA se ha completado. Tal como será conocido por los expertos en la materia, una memoria física que es adecuada para un DMA en general no se diseña en detalle durante el funcionamiento normal del dispositivo informático. No obstante, la memoria dentro del entorno de máquina virtual se diseña en detalle casi siempre, habitualmente por medio del hipervisor. En consecuencia, puede que sea necesario proteger las direcciones de memoria que se pasan a la extensión de una forma que normalmente no necesitaría ser realizada con la memoria que se atribuye a otros procesos en el entorno de máquina virtual. Tal protección puede ser realizada por el hipervisor, que puede usar un mecanismo que comúnmente se conoce como “anclaje” para “fijar” las ubicaciones de memoria especificadas hasta que se haya completado el DMA.

Resulta obvio que, una vez que un DMA se ha completado, el hipervisor puede liberar, o “desanclar”, las ubicaciones de memoria especificadas. La compleción de un DMA se puede detectar de una forma muy similar a aquella en la que se podría detectar un DMA próximo, lo que se explicó con detalle en lo que antecede. Por ejemplo, el hipervisor 613 o el proceso de sistema operativo virtual 611 podría supervisar las funciones que son invocadas por la extensión 615. Las funciones tales como una desatribución de las ubicaciones de memoria especificadas pueden indicar que el DMA se ha completado, y se pueden usar como una indicación de que el hipervisor 613 puede desanclar las ubicaciones de memoria especificadas.

Un procedimiento adicional de comunicación directa con hardware que se aborda por medio de algunas formas de realización de la presente invención se refiere a la entrega de las interrupciones de hardware a un código que se está ejecutando dentro de un entorno de máquina virtual. Tal como será conocido por los expertos en la materia, una interrupción de hardware puede ser una señal a partir de un dispositivo de hardware, que se envía a un controlador de dispositivo apropiado u otra aplicación de software, que requiere, en general, un cierto tipo de respuesta o acuse de recibo. Debido a que, tal como se ha descrito en lo que antecede, el sistema operativo de host puede no ser capaz de soportar el controlador de dispositivo correcto, u otro software de control, para un dispositivo de hardware particular, puede ser necesario que la interrupción se dirija a una extensión que se está ejecutando en el interior de un entorno de máquina virtual. Por ejemplo, el dispositivo informático 100 de la figura 1 se muestra conectado a un dispositivo heredado 199. Si el sistema operativo 134 es un sistema operativo moderno, este puede no ser capaz de soportar de forma correcta un controlador de dispositivo para el dispositivo heredado 199. Por lo tanto, para posibilitar que un usuario del dispositivo informático 100 use el dispositivo heredado 199, un controlador de dispositivo, o un software de control similar, se puede ejecutar dentro de un entorno virtual. En consecuencia, cualquier interrupción que se reciba a partir del dispositivo heredado 199 solo puede ser manejada de forma correcta si la misma se dirige al proceso de máquina virtual, y se las permite pasar a su través hasta el controlador de

dispositivo.

Un mecanismo para dirigir las interrupciones a una extensión, tal como la extensión 615, que se contempla por medio de una forma de realización de la presente invención requiere que una interrupción recibida se compare con una tabla, o con una construcción similar, para determinar si el proceso de máquina virtual 617 debería manejar la interrupción o pasar la misma al proceso de sistema operativo de host 601. Más en concreto, en un dispositivo informático que tiene solo una única CPU, las interrupciones se pueden recibir o bien cuando el proceso de máquina virtual 617 se está ejecutando en la CPU, o bien cuando el proceso de sistema operativo de host 601 se está ejecutando en la CPU. El presente mecanismo se puede aplicar a la situación en la que la interrupción llega mientras el proceso de máquina virtual 617 se está ejecutando en la CPU. En un caso de ese tipo, el hipervisor 613 puede determinar la razón o el destino de la interrupción. El hipervisor 613 puede determinar entonces si la interrupción es manejada de forma apropiada por una extensión en el entorno de máquina virtual, tal como la extensión 615, al, por ejemplo, llevar a cabo una consulta en una tabla. Si la interrupción es manejada de forma apropiada por la extensión 615, el hipervisor 613 puede pasar la interrupción al proceso de máquina virtual 617 y, por lo tanto, a la extensión. Si la interrupción es manejada de forma apropiada por una extensión u otra aplicación de software que está asociada con el proceso de sistema operativo de host 601, el hipervisor 613 puede completar la ejecución del proceso de máquina virtual 617 en el hardware 620, y permitir que el proceso de sistema operativo de host reanude la ejecución en el hardware, y que realice la interrupción de una forma apropiada.

Si el hipervisor 613 pasa la interrupción al proceso de máquina virtual 617, este puede modificar el número de la línea de interrupción en la que llegó la interrupción, con el fin de mantener la compatibilidad con el proceso de sistema operativo virtual 611. Por lo tanto, cuando se habilita una línea de interrupción, el hipervisor 613 puede verificar que la información de línea de interrupción se corresponde con una línea de interrupción física. El hipervisor 613 puede traducir entonces entre la línea de interrupción física y una línea de interrupción emulada.

Debido a que una máquina virtual puede emular un hardware que es diferente del hardware 620 en el que se está ejecutando el proceso de máquina virtual 617, el hipervisor 613 puede necesitar emular una única instrucción de máquina virtual como múltiples instrucciones en el hardware de host. Por ejemplo, si una máquina virtual está emulando un tipo diferente de CPU que la CPU física en la que se está ejecutando la misma, las instrucciones que pueden requerir solo un único ciclo de CPU cuando se llevan a cabo por la CPU que se está emulando pueden requerir múltiples ciclos de CPU cuando se llevan a cabo por la CPU física. En un caso de ese tipo, puede ser importante que el hipervisor 613 trate los múltiples ciclos de CPU de la CPU física de una forma unitaria con el fin de mantener la compatibilidad con la CPU emulada. Por lo tanto, si una interrupción de hardware llega mientras el hipervisor 613 se encuentra en medio de la ejecución de una serie de ciclos en la CPU física que están correlacionados con un único ciclo de la CPU emulada, el hipervisor puede ignorar, poner en cola o retardar de otro modo la interrupción hasta que se ha completado la serie de ciclos de CPU.

Mecanismos adicionales para dirigir las interrupciones a una extensión en un proceso de máquina virtual que se contempla por medio de una forma de realización de la presente invención solicitan que el proceso de sistema operativo de host o bien retarde la interrupción antes de transferir el control al proceso de máquina virtual, o bien transfiera el control al proceso de máquina virtual tan pronto como se recibe la interrupción, o bien intente ejecutar la extensión dentro del proceso de host con unos punteros apropiados al proceso de máquina virtual. Tal como se ha explicado en lo que antecede, en un dispositivo informático que tiene solo una única CPU, las interrupciones se pueden recibir o bien cuando el proceso de máquina virtual 617 se está ejecutando en la CPU, o bien cuando el proceso de sistema operativo de host 601 se está ejecutando en la CPU. Los presentes mecanismos se pueden aplicar a la situación en la que la interrupción llega mientras el proceso de sistema operativo de host 601 se está ejecutando en la CPU. Como cuestión inicial, es probable que el sistema operativo de host haya definido previamente unos procedimientos para dirigir la interrupción a los controladores de dispositivo apropiados. Tales procedimientos se pueden establecer, por ejemplo, durante el proceso de arranque del sistema operativo de host, tal como cuando se cargan los controladores de dispositivo. La invocación de la extensión 615 puede intentar, por lo tanto, aprovecharse de estos procedimientos previamente definidos e indicar al proceso de sistema operativo de host 601 que las interrupciones que se reciben a partir de un dispositivo de hardware particular se deberían dirigir al proceso de máquina virtual 617.

En consecuencia, cuando una interrupción que se debería enviar a la extensión 615 se recibe mientras el proceso de sistema operativo de host 601 se está ejecutando en la CPU, el proceso de sistema operativo de host puede llevar a cabo unos procedimientos que son similares a los que se ponen en práctica cuando este recibe cualquier otra interrupción, con la excepción de que el mismo puede determinar que el software apropiado para manejar la interrupción se está ejecutando dentro del proceso de máquina virtual 617. El proceso de sistema operativo de host 601 puede intentar entonces transferir la interrupción a la extensión 615 al, por ejemplo, deshabilitar las interrupciones, completar una o más tareas, conmutar la ejecución al proceso de máquina virtual 617 y, a continuación, volver a habilitar las interrupciones. Debido a que el proceso de máquina virtual 617 se estará ejecutando, por lo tanto, en la CPU cuando se vuelven a habilitar las interrupciones, la interrupción se puede recibir por medio del proceso de máquina virtual 617 y puede ser manejada por la misma de la forma que se ha descrito con detalle en lo que antecede.

Tal como será conocido por los expertos en la materia, en general los dispositivos de hardware pueden usar dos tipos diferentes de interrupciones: una interrupción permanente que permanece activa hasta que la misma ha sido abordada, o se ha respondido a la misma, y una interrupción transitoria que puede producir un bloqueo temporal y, a continuación, terminar. Usando el mecanismo anteriormente descrito, el proceso de máquina virtual 617 puede detectar una interrupción permanente tan pronto como se vuelven a habilitar las interrupciones, debido a que la interrupción permanente nunca se desactivó. Por lo tanto, para una interrupción permanente, el proceso de máquina virtual 617 puede usar los mecanismos que se han descrito con detalle en lo que antecede para manejar la interrupción de la misma forma que si la misma hubiera llegado en un primer momento mientras que el proceso de máquina virtual se estaba ejecutando en la CPU. Para una interrupción transitoria, no obstante, el bloqueo temporal, que puede indicar que ha tenido lugar una interrupción, puede quedar deshecho. En consecuencia, a menos que tenga lugar otra interrupción para volver a producir el bloqueo temporal, el proceso de máquina virtual 617 nunca puede aprender acerca de la interrupción si la misma tuvo lugar mientras que el proceso de sistema operativo de host 601 se estaba ejecutando en la CPU. Por lo tanto, el proceso de sistema operativo de host 601 puede realizar un seguimiento de o almacenar de otro modo una o más interrupciones transitorias que tienen lugar antes de la transferencia de la ejecución al proceso de máquina virtual 617. El proceso de sistema operativo de host 601 puede pasar información al hipervisor 613 para informar al hipervisor de que ha tenido lugar una interrupción transitoria, y puede proporcionar el número de interrupciones transitorias, si es apropiado. Una vez que el proceso de máquina virtual 617 se está ejecutando en la CPU, el hipervisor 613 puede emular entonces las interrupciones transitorias una tras otra, y permitir que la extensión 615 responda a las mismas del mismo modo. Una vez que el hipervisor 613 ha completado la emulación de las interrupciones transitorias, este puede volver a habilitar entonces las interrupciones.

En algunos casos, puede ser necesario que las interrupciones de hardware sean manejadas, o que se responda a las mismas, con una velocidad mayor que la que pueden proporcionar los procedimientos anteriores. En un caso de ese tipo, un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que el proceso de sistema operativo de host 601 transfiera inmediatamente la ejecución al proceso de máquina virtual 617 cuando se detecta una interrupción que es manejada de forma correcta por una extensión que se está ejecutando en el proceso de máquina virtual, tal como la extensión 615, en lugar de deshabilitar las interrupciones e intentar completar una o más tareas usando los mecanismos anteriormente descritos. No obstante, el hipervisor 613 puede ser de tipo subproceso único, lo que puede retardar la detección de la interrupción y, en consecuencia, el servicio de la interrupción, si el hipervisor está aguardando una respuesta o alguna otra información.

Para evitar un retardo debido a la naturaleza de tipo subproceso único de un hipervisor, una variante del mecanismo anterior que también se contempla por medio de una forma de realización de la presente invención solicita que el hipervisor 613 emule un dispositivo informático de múltiples CPU y que el proceso de sistema operativo virtual 611 sea capaz de operar en un entorno de múltiples CPU. Además, el hipervisor 613 puede estructurar la ejecución de instrucciones de una forma tal que al menos una CPU emulada se conserva en un estado en el que la misma puede aceptar interrupciones. Por ejemplo, tal como se ha descrito en lo que antecede, el proceso de máquina virtual 617 puede ser llamado a partir del proceso de sistema operativo de host 601 al pasar una orden al proceso de máquina virtual y, a continuación, almacenar en memoria caché los apuntalamientos del proceso de sistema operativo de host y ejecutar el proceso de máquina virtual en el hardware 620. El hipervisor 613 puede conservar una CPU emulada en un estado en el que la misma puede aceptar interrupciones al pasar las órdenes que se reciben a partir del proceso de sistema operativo de host 601 a otra CPU emulada. En consecuencia, debido a que no se permite que la CPU maneje unas órdenes a partir del proceso de sistema operativo de host 601, la misma puede mantener un estado en el que esta puede manejar inmediatamente una interrupción recibida.

En consecuencia, si una interrupción fuera a llegar mientras que el proceso de sistema operativo de host subyacente 601 se estaba ejecutando en el hardware 620, y la interrupción requiere una latencia baja, el proceso de sistema operativo de host puede transferir el control al proceso de máquina virtual 617 tan rápidamente como sea posible. Una vez que el proceso de máquina virtual 617 ha comenzado a ejecutarse en el hardware 620, al menos una CPU emulada del proceso de máquina virtual se encuentra en un estado en el que la misma puede aceptar la interrupción. Por lo tanto, incluso si otras CPU emuladas se encontraran en un estado en el que las mismas estuvieran llevando a cabo una función, o aguardando una respuesta, la interrupción puede ser manejada de una forma eficiente por la al menos una CPU emulada que se reservó para las interrupciones. El hipervisor 613 y el proceso de sistema operativo virtual 611 pueden poner en práctica entonces las etapas necesarias para entregar la interrupción al software apropiado, tal como la extensión 615, de la forma que se ha descrito con detalle en lo que antecede. Además, debido a que el hipervisor 613 puede requerir que se ancle la memoria física, tal como también se ha descrito en lo que antecede, se puede permitir que la CPU emulada que recibió la interrupción complete el manejo de la interrupción antes de devolver el control a otra CPU emulada o a otro proceso. De tal forma, se puede reservar al menos una CPU emulada para un manejo inmediato de las interrupciones.

Otro mecanismo que proporciona un manejo de latencia baja de las interrupciones de hardware que se contempla por medio de una forma de realización de la presente invención solicita que el proceso de sistema operativo de host 601 capture el código para una rutina de servicio de interrupción a partir de la extensión 615 y que ejecute el propio código, con unos punteros de datos apropiados de vuelta al proceso de máquina virtual 617. Por ejemplo, el proceso de sistema operativo de host 601 puede seguir el rastro de las rutinas de servicio de interrupción apropiadas desde el comienzo del espacio de memoria del proceso de máquina virtual 617. Una vez que se han localizado, esas

rutinas de servicio de interrupción se pueden copiar en el proceso de sistema operativo de host 601 y ejecutarse ahí con el fin de manejar la interrupción con una latencia muy baja.

Debido a que se tenía por objeto que las rutinas de servicio de interrupción se ejecutaran dentro del espacio de proceso del proceso de máquina virtual 617, el proceso de sistema operativo de host 601, cuando este copia esas rutinas y las ejecuta, puede proporcionar unos punteros de datos de vuelta al proceso de máquina virtual de tal modo que las rutinas pueden operar de forma correcta. Por ejemplo, el proceso de sistema operativo de host 601 puede cambiar las instrucciones apropiadas de las rutinas de servicio de interrupción, o las asignaciones de tabla de páginas, para hacer referencia a la memoria dentro del proceso de máquina virtual 617. Las técnicas conocidas de aislamiento frente a fallos de software se pueden usar para modificar las instrucciones apropiadas, y para proporcionar una medida del aislamiento frente a fallos. Tal como será conocido por los expertos en la materia, la ejecución de software se puede supervisar mediante la inserción de unas órdenes apropiadas entre las órdenes del software que se está supervisando. Para evitar la necesidad de volver a compilar el software que se está supervisando, las órdenes insertadas pueden ser unas órdenes de bajo nivel que se pueden insertar en un código compilado. Por ejemplo, una instrucción de bajo nivel para acceder a una ubicación de memoria particular al copiar los contenidos de esa ubicación en un registro de un procesador se puede ver precedida por una instrucción insertada que comprueba la dirección de la ubicación de memoria a la que se está accediendo, tal como mediante la comparación de la dirección con un intervalo conocido de direcciones. Si la ubicación de memoria es una ubicación incorrecta, por ejemplo, si la misma se encuentra fuera de un intervalo de direcciones apropiado, se puede hacer una modificación para sustituir una dirección apropiada en la solicitud de acceso. De tal forma, cada instrucción de acceso a memoria se puede modificar para acceder a una ubicación de memoria correcta, a pesar del hecho de que la rutina de manejo de interrupción se puede estar ejecutando en el proceso de sistema operativo de host 601 en lugar del proceso de máquina virtual 617.

Tal como se indica, las técnicas de aislamiento frente a fallos de software también pueden proporcionar una medida del aislamiento frente a fallos a pesar de la ejecución de las rutinas de manejo de interrupción directamente en el proceso de sistema operativo de host 601. Por ejemplo, un aspecto de aislamiento frente a fallos de software se logra mediante la inserción de unas instrucciones de bajo nivel antes de cada instrucción de escritura en memoria para asegurar que la ubicación a la que se dirige la instrucción de escritura es una ubicación correcta. Tal como será conocido por los expertos en la materia, los fallos de software a menudo dan lugar a inestabilidad debido a que el fallo dio como resultado que se escribieran unos datos en una ubicación de memoria incorrecta. Además, tales instrucciones de escritura incorrectas pueden ser difíciles de detectar debido a que la dirección en la que se deberían escribir los datos puede no determinarse hasta la compleción de la instrucción inmediatamente precedente. Mediante la inserción de las instrucciones que se han descrito en lo que antecede inmediatamente antes de cualquier escritura en memoria, las direcciones de memoria a las que se dirigen tales instrucciones de escritura se pueden comprobar, tal como, por ejemplo, mediante la comparación de las mismas con un intervalo conocido de direcciones de memoria. Una indicación de que la escritura se dirige hacia una ubicación de memoria en el exterior del intervalo conocido puede indicar, por lo tanto, que la instrucción de escritura es incorrecta y puede dar lugar a inestabilidad. En consecuencia, la instrucción de escritura se puede modificar o abortarse, y se puede lograr una medida del aislamiento frente a fallos. También se pueden usar aspectos adicionales de aislamiento frente a fallos de software, incluyendo un flujo de control de espacio asilado, el uso de unas instrucciones privilegiadas, y similares. Se puede hallar información adicional con respecto a los diversos aspectos de aislamiento frente a fallos de software, incluyendo los que se han descrito en lo que antecede, en la patente de EE. UU. con n.º 5.761.477 a nombre de Wahbe y col., cuyos contenidos se incorporan en el presente documento por referencia en su totalidad para explicar o describir adicionalmente cualquier enseñanza o sugerencia que esté contenida dentro de la presente memoria descriptiva que sea consistente con sus divulgaciones.

No obstante, determinados dispositivos informáticos pueden tener múltiples CPU físicas, caso en el cual algunos de los mecanismos anteriores pueden no ser necesarios. Por ejemplo, en un dispositivo informático con múltiples CPU físicas, una única CPU física puede estar ejecutando siempre el proceso de máquina virtual 617. En un caso de ese tipo, un mecanismo que se contempla por medio de una forma de realización de la presente invención solicita que el mecanismo de control de las interrupciones de hardware, que a menudo puede ser un conjunto de circuitos dedicados que es parte del propio dispositivo informático, dirija todas las interrupciones que requieren una extensión, tal como la extensión 615, para que se dirijan a la CPU física en la que siempre se está ejecutando el proceso de máquina virtual 617. Incluso si el proceso de máquina virtual 617 comparte una CPU física con otros procesos, pero comparte siempre la misma CPU física, dirigir todas las interrupciones que requieren la extensión 615 a esa CPU física puede seguir proporcionando una solución óptima cuando se combina con los mecanismos anteriormente descritos para transferir interrupciones al proceso de máquina virtual apropiado, incluso si el mismo no se está ejecutando actualmente en la CPU física.

No obstante, si el proceso de máquina virtual 617 se puede estar ejecutando en una cualquiera de las múltiples CPU físicas, entonces se pueden usar unos mensajes entre procesadores para permitir que cualquier procesador responda a una interrupción de hardware. Por ejemplo, si ocurre que el proceso de máquina virtual 617 se está ejecutando en una primera CPU física y una interrupción llega a una segunda CPU física que puede ser manejada por la extensión 615, la segunda CPU física puede comunicar la información relevante a la primera CPU física para permitir que la extensión maneje la interrupción de hardware. Tal como será conocido por los expertos en la materia, puede ser muy difícil reenviar físicamente una interrupción de hardware de una CPU física a otra. En consecuencia,

mediante el uso de mensajes entre procesadores, la interrupción se puede manejar como si la misma hubiera llegado a la CPU física correcta.

A la vista de las muchas formas de realización posibles a las que se pueden aplicar los principios de la presente invención, se debería reconocer que las formas de realización que se describen en el presente documento con respecto a las figuras de dibujo tienen por objeto ser únicamente ilustrativas y no se deberían interpretar como limitantes del ámbito de la invención. Por ejemplo, los expertos en la materia reconocerán que algunos elementos de las formas de realización ilustradas que se muestran en software se pueden poner en práctica en hardware y viceversa, o que las formas de realización que se ilustran se pueden modificar en cuanto a su disposición. De forma similar, se debería reconocer que los mecanismos que se describen en el contexto de un entorno de máquina virtual pueden ser aplicables al entorno virtual que se crea encima de un sistema operativo común, y viceversa. Por ejemplo, las técnicas de aislamiento frente a fallos de software que se han descrito en lo que antecede junto con los entornos de máquina virtual se pueden aplicar de igual modo a cualquier situación en la que pueda ser no deseable una conmutación de contexto excesiva, incluyendo rutinas de extensión que se copian de un proceso virtual a un proceso de host incluso cuando ambos procesos comparten un sistema operativo subyacente común. Por lo tanto, la invención tal como se describe en el presente documento contempla la totalidad de tales formas de realización según puedan entrar dentro del ámbito de las siguientes reivindicaciones y equivalentes de las mismas.

REIVINDICACIONES

1. Un procedimiento de aislamiento de una extensión usada por un proceso de host, comprendiendo el procedimiento las etapas de:

detectar una operación por el proceso de host, en el que la operación está relacionada con una funcionalidad de la extensión;
identificar una entidad representante para la extensión, en el que la entidad representante soporta la funcionalidad relacionada con la operación;
cargar la entidad representante en el proceso de host;
cargar la extensión en un proceso virtual, en el que el proceso virtual es una instancia virtual del proceso de host; en el que la operación es una invocación de la extensión, y en el que la entidad representante expone, al proceso de host, una API de servicio que está asociada con la extensión y el proceso virtual proporciona, a la extensión, una API de soporte que está asociada con el proceso de host;
reenviar desde la entidad representante en el proceso de host a la extensión en el proceso virtual una solicitud que está asociada con la operación; y
reenviar, a la entidad representante en el proceso de host, una respuesta a partir de la extensión, en el que la respuesta puede ser usada por la entidad representante para soportar la operación por el proceso de host;
comprendiendo adicionalmente el procedimiento:

ajustarse de forma rigurosa, la entidad representante, a las API de servicio presentadas por la extensión;
si la extensión intenta devolver datos al proceso de host que no es de la forma o el tipo que el host está esperando, identificar, la entidad representante, el problema potencial y no pasar esos datos al proceso de host;
detectar un error en el proceso virtual; y
devolver el control al proceso de host con una indicación de error.

2. El procedimiento de la reivindicación 1, en el que el reenvío de la solicitud comprende reenviar, a un código auxiliar en el proceso virtual, la solicitud a partir del proceso de host, en el que la extensión recibe la solicitud a partir del código auxiliar de la misma forma en la que la extensión hubiera recibido la solicitud a partir del proceso de host si la extensión se hubiera cargado en el proceso de host.

3. El procedimiento de la reivindicación 1, en el que el reenvío de la respuesta comprende: detectar una respuesta no conforme o un fallo de la extensión o el proceso virtual; y proporcionar al proceso de host, por la entidad representante, una indicación apropiada que habilita una degradación correcta de la funcionalidad que es proporcionada por el proceso de host.

4. El procedimiento de la reivindicación 1, que comprende adicionalmente las etapas de: proporcionar, en el proceso virtual, un contexto de modo de usuario del proceso de host al copiar el contexto de modo de usuario a partir del proceso de host en el proceso virtual.

5. El procedimiento de la reivindicación 1, que comprende adicionalmente las etapas de:

proporcionar, en el proceso virtual, un contexto de modo de usuario del proceso de host mediante la asignación de una memoria de proceso de host y una memoria de proceso virtual a una memoria física común; y
proteger la memoria física común frente al acceso de escritura por el proceso virtual.

6. El procedimiento de la reivindicación 1, en el que el proceso virtual es un proceso de máquina virtual que se crea usando un estado coherente guardado, creado el estado coherente guardado al llevar a cabo unas etapas que comprenden: arrancar una máquina virtual inicial; y guardar un estado coherente que es creado por el arranque de la máquina virtual inicial.

7. El procedimiento de la reivindicación 1, en el que el proceso de host es un proceso de sistema operativo y el proceso virtual es un proceso de máquina virtual que se crea usando un estado coherente guardado, creado el estado coherente guardado mediante el registro de un estado durante un arranque del proceso de sistema operativo.

8. Un medio legible por ordenador que tiene unas instrucciones ejecutables por ordenador para aislar una extensión usada por un proceso de host, comprendiendo el medio legible por ordenador unas instrucciones ejecutables por ordenador para:

detectar una operación por el proceso de host, en el que la operación está relacionada con una funcionalidad de la extensión;
identificar una entidad representante para la extensión, en el que la entidad representante soporta la funcionalidad relacionada con la operación;
cargar la entidad representante en el proceso de host;
cargar la extensión en un proceso virtual, en el que el proceso virtual es una instancia virtual del proceso de host; en el que la operación es una invocación de la extensión, y en el que la entidad representante expone, al proceso de host, una API de servicio que está asociada con la extensión y el proceso virtual proporciona, a la extensión,

una API de soporte que está asociada con el proceso de host;
reenviar desde la entidad representante en el proceso de host a la extensión en el proceso virtual una solicitud que está asociada con la operación; y
reenviar, a la entidad representante en el proceso de host, una respuesta a partir de la extensión, en el que la respuesta puede ser usada por la entidad representante para soportar la operación por el proceso de host;
ajustarse de forma rigurosa, la entidad representante, a las API de servicio presentadas por la extensión;
si la extensión intenta devolver datos al proceso de host que no es de la forma o el tipo que el host está esperando, identificar, la entidad representante, el problema potencial y no pasar esos datos al proceso de host;
detectar un error en el proceso virtual; y
devolver el control al proceso de host con una indicación de error.

9. El medio legible por ordenador de la reivindicación 8, en el que el proceso de host no está diseñado para invocar la extensión.

10. El medio legible por ordenador de la reivindicación 8, que comprende adicionalmente unas instrucciones ejecutables por ordenador para ejecutar una función ligera comúnmente invocada de la extensión en el proceso de host usando técnicas de aislamiento frente a fallos de software.

11. El medio legible por ordenador de la reivindicación 8, en el que las instrucciones ejecutables por ordenador para reenviar la solicitud comprenden unas instrucciones ejecutables por ordenador para reenviar, a un código auxiliar en el proceso virtual, la solicitud a partir del proceso de host, en el que la extensión recibe la solicitud a partir del código auxiliar de la misma forma en la que la extensión hubiera recibido la solicitud a partir del proceso de host si la extensión se hubiera cargado en el proceso de host.

12. El medio legible por ordenador de la reivindicación 8, en el que las instrucciones ejecutables por ordenador para reenviar la respuesta comprenden unas instrucciones ejecutables por ordenador para: detectar una respuesta no conforme o un fallo de la extensión o el proceso virtual; y proporcionar al proceso de host, por la entidad representante, una indicación apropiada que habilita una degradación correcta de la funcionalidad que es proporcionada por el proceso de host.

13. El medio legible por ordenador de la reivindicación 8, que comprende adicionalmente unas instrucciones ejecutables por ordenador para: proporcionar, en el proceso virtual, un contexto de modo de usuario del proceso de host al copiar el contexto de modo de usuario a partir del proceso de host en el proceso virtual.

14. El medio legible por ordenador de la reivindicación 8, que comprende adicionalmente unas instrucciones ejecutables por ordenador para:

proporcionar, en el proceso virtual, un contexto de modo de usuario del proceso de host mediante la asignación de una memoria de proceso de host y una memoria de proceso virtual a una memoria física común; y
proteger la memoria física común frente al acceso de escritura por el proceso virtual.

15. El medio legible por ordenador de la reivindicación 14, en el que la protección de la memoria física común comprende implementar un esquema de protección de memoria que está seleccionado de entre un grupo de esquemas de protección de memoria que comprende: acceso de solo lectura, acceso de copia en escritura y acceso de escritura no simultánea examinada.

16. El medio legible por ordenador de la reivindicación 8, en el que el proceso virtual es un proceso de máquina virtual que se crea usando un estado coherente guardado, creado el estado coherente guardado por unas instrucciones ejecutables por ordenador para llevar a cabo unas etapas que comprenden: arrancar una máquina virtual inicial; y guardar un estado coherente que es creado por el arranque de la máquina virtual inicial.

17. El medio legible por ordenador de la reivindicación 8, en el que el proceso de host es un proceso de sistema operativo y el proceso virtual es un proceso de máquina virtual que se crea usando un estado coherente guardado, creado el estado coherente guardado mediante el registro de un estado durante un arranque del proceso de sistema operativo.

18. Un dispositivo informático que comprende: uno o más procesadores; y un almacenamiento en memoria, comprendiendo el almacenamiento en memoria unas instrucciones ejecutables por ordenador para:

crear un proceso de host, en el que el proceso de host lleva a cabo una operación;
crear un proceso virtual, en el que el proceso virtual es una instancia virtual del proceso de host, y en el que adicionalmente el proceso virtual soporta una extensión que está asociada con la operación;
cargar una entidad representante que está asociada con la operación en el proceso de host;
cargar la extensión en el proceso virtual;
en el que la operación es una invocación de la extensión, y en el que la entidad representante expone, al proceso de host, una API de servicio que está asociada con la extensión y el proceso virtual proporciona, a la extensión, una API de soporte que está asociada con el proceso de host;
reenviar, a la extensión, una solicitud a partir de la entidad representante; y

reenviar, a la entidad representante, una respuesta a partir de la extensión, en el que la respuesta puede ser usada por la entidad representante para soportar la operación;
ajustarse de forma rigurosa, la entidad representante, a las API de servicio presentadas por la extensión;
si la extensión intenta devolver datos al proceso de host que no es de la forma o el tipo que el host está esperando, identificar, la entidad representante, el problema potencial y no pasar esos datos al proceso de host;
detectar un error en el proceso virtual; y
devolver el control al proceso de host con una indicación de error.

19. El dispositivo informático de la reivindicación 18, en el que el almacenamiento en memoria comprende adicionalmente unas instrucciones ejecutables por ordenador para: detectar una respuesta no conforme o un fallo de la extensión o el proceso virtual; y proporcionar al proceso de host, por la entidad representante, una indicación apropiada que habilita una degradación correcta de la funcionalidad que es proporcionada por el proceso de host.

20. El dispositivo informático de la reivindicación 18, en el que el almacenamiento en memoria comprende adicionalmente unas instrucciones ejecutables por ordenador para: proporcionar, en el proceso virtual, un contexto de modo de usuario del proceso de host al copiar el contexto de modo de usuario a partir del proceso de host en el proceso virtual o mediante la asignación de una memoria de proceso de host y una memoria de proceso virtual a una memoria física común.

21. El dispositivo informático de la reivindicación 18, en el que el proceso de host es un proceso de sistema operativo y el proceso virtual es un proceso de máquina virtual que se crea usando un estado coherente guardado, creado el estado coherente guardado por unas instrucciones ejecutables por ordenador que ponen en práctica unas etapas que comprenden: arrancar una máquina virtual inicial y guardar un estado que es creado por el arranque de la máquina virtual inicial; o por unas instrucciones ejecutables por ordenador que ponen en práctica unas etapas que comprenden: guardar un estado que es creado por un arranque del proceso de sistema operativo.

22. El dispositivo informático de la reivindicación 18, en el que el proceso de host es un proceso de sistema operativo y el proceso virtual es un proceso de máquina virtual, el almacenamiento en memoria comprende adicionalmente unas instrucciones ejecutables por ordenador para:

indicar, durante un arranque del proceso de sistema operativo, que el dispositivo informático tiene uno o más procesadores adicionales;
guardar un estado durante el arranque del proceso de sistema operativo después de que se hayan inicializado los uno o más procesadores y los uno o más procesadores adicionales;
indicar al proceso de sistema operativo, después de que al menos un enlace de componente que está asociado con los uno o más procesadores haya sido completado por el proceso de sistema operativo, que los uno o más procesadores adicionales han fallado;
proporcionar, al proceso de máquina virtual, el estado guardado;
arrancar el proceso de máquina virtual usando el estado guardado, en el que el arranque del proceso de máquina virtual comprende evitar que los uno o más procesadores accedan a al menos un componente que está asociado con el al menos un enlace de componente; y
proporcionar, al proceso de máquina virtual, un acceso al al menos un enlace de componente usando un soporte preexistente para proporcionar, a los uno o más procesadores adicionales, un acceso al al menos un enlace de componente que está asociado con los uno o más procesadores.

23. El dispositivo informático de la reivindicación 22, en el que el almacenamiento en memoria comprende adicionalmente unas instrucciones ejecutables por ordenador para: crear un enlace de componente especificado en el proceso de máquina virtual con un componente especificado, en el que el componente especificado interacciona con un controlador de dispositivo alojado dentro del proceso de máquina virtual.

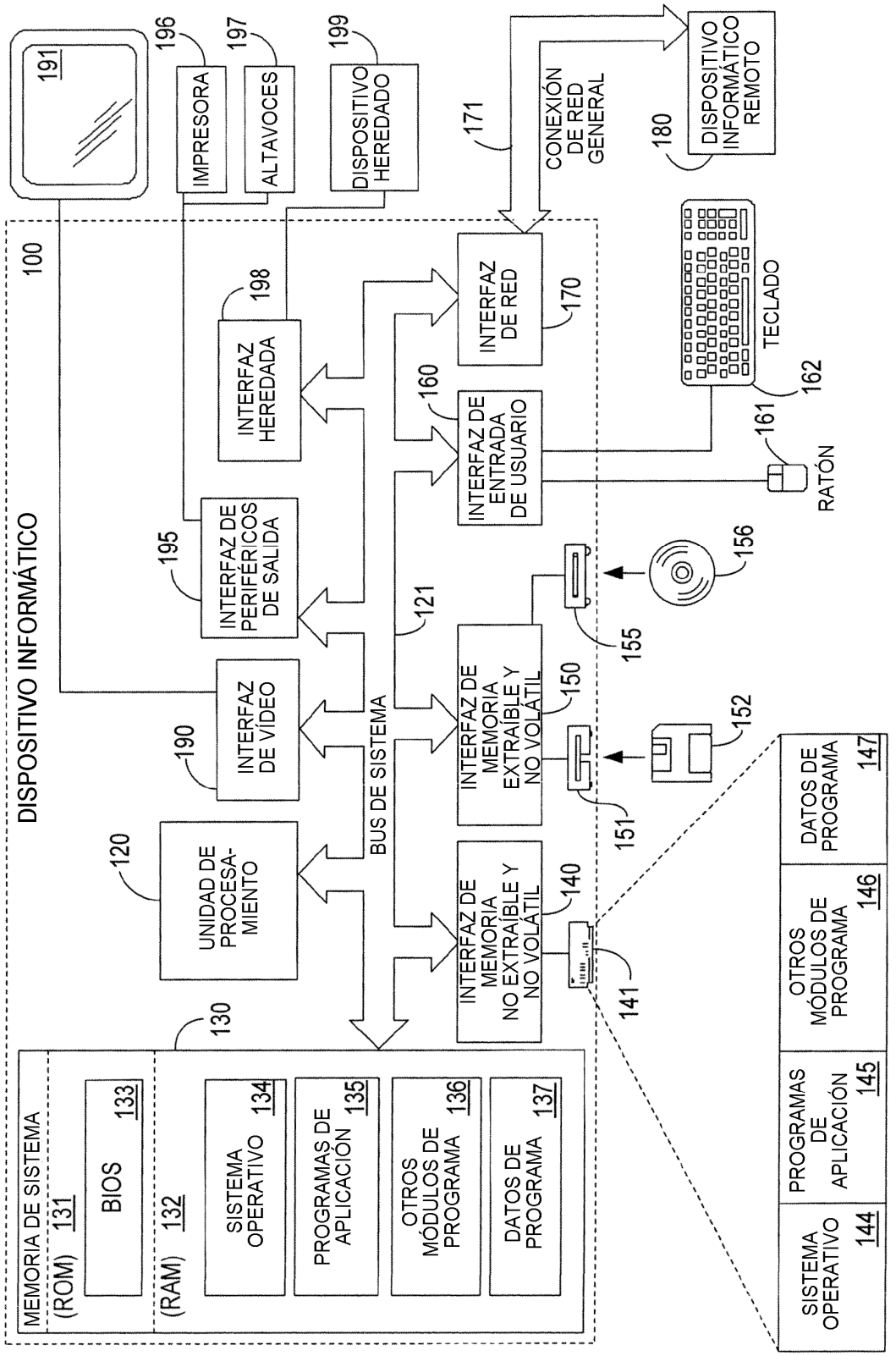
DISPOSITIVO INFORMÁTICO
100
MEMORIA DE SISTEMA
(ROM) 131
BIOS 133
(RAM) 132
SISTEMA OPERATIVO 134
PROGRAMAS DE APLICACIÓN 135
OTROS MÓDULOS DE PROGRAMA 136
DATOS DE PROGRAMA 137
130
UNIDAD DE PROCESAMIENTO
120
BUS DE SISTEMA
121
INTERFAZ DE VÍDEO
190
INTERFAZ DE PERIFÉRICOS DE SALIDA
195
INTERFAZ HEREDADA
198
INTERFAZ DE MEMORIA NO EXTRAÍBLE Y NO VOLÁTIL
140
INTERFAZ DE MEMORIA EXTRAÍBLE Y NO VOLÁTIL
150
INTERFAZ DE ENTRADA DE USUARIO
160
INTERFAZ DE RED
170
191
IMPRESORA
196
ALTAVOCES
197
DISPOSITIVO HEREDADO
199
CONEXIÓN DE RED GENERAL
171
DISPOSITIVO INFORMÁTICO REMOTO
180
TECLADO
162
RATÓN
161
156
155
152
151
141
SISTEMA OPERATIVO 144
PROGRAMAS DE APLICACIÓN 145
OTROS MÓDULOS DE PROGRAMA 146
DATOS DE PROGRAMA 147

Figura 1

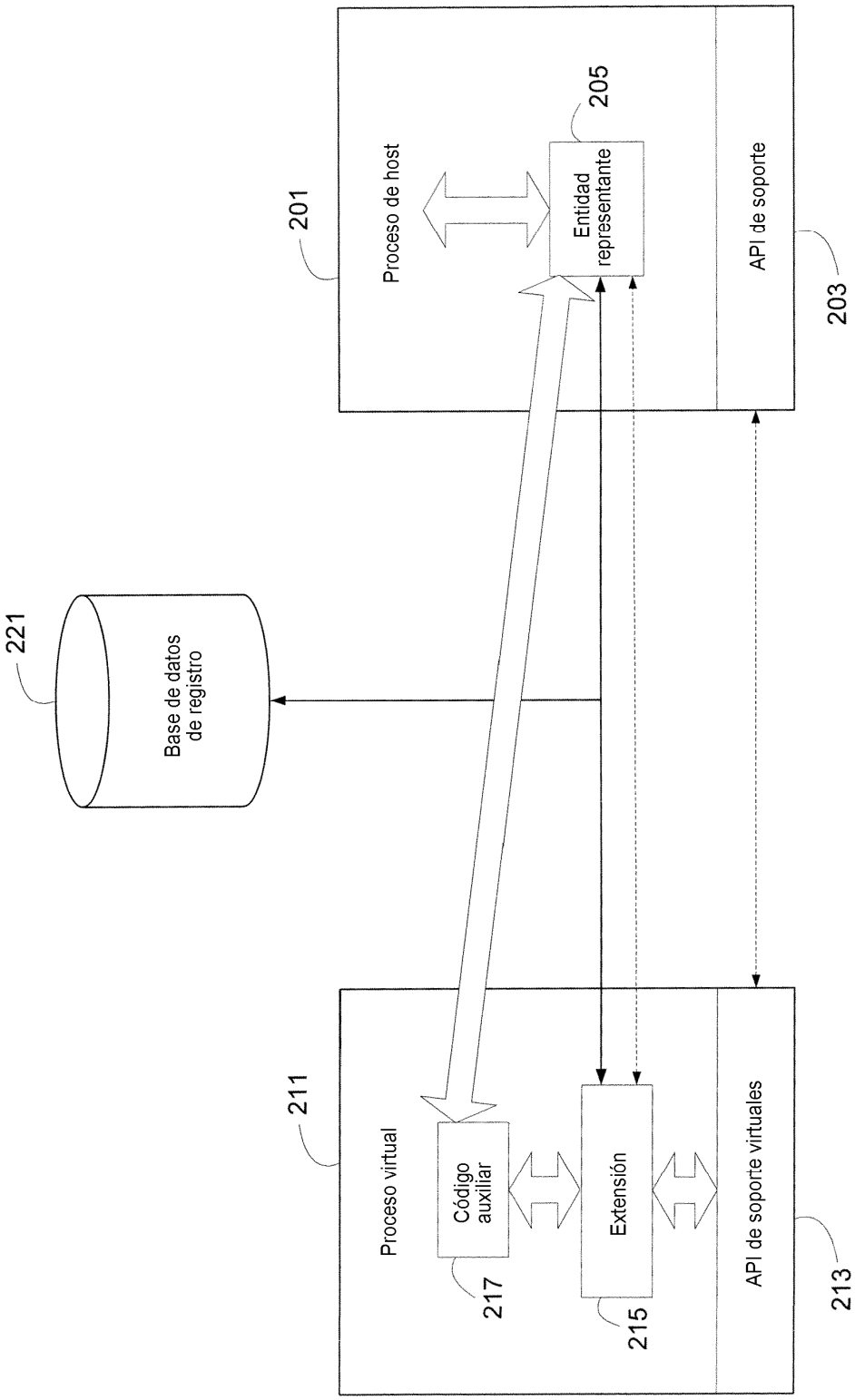
201
Proceso de host
205
Entidad representante
API de soporte
203
221
Base de datos de registro
211
Proceso virtual
Código auxiliar
217
Extensión
215
API de soporte virtuales
213

Figura 2

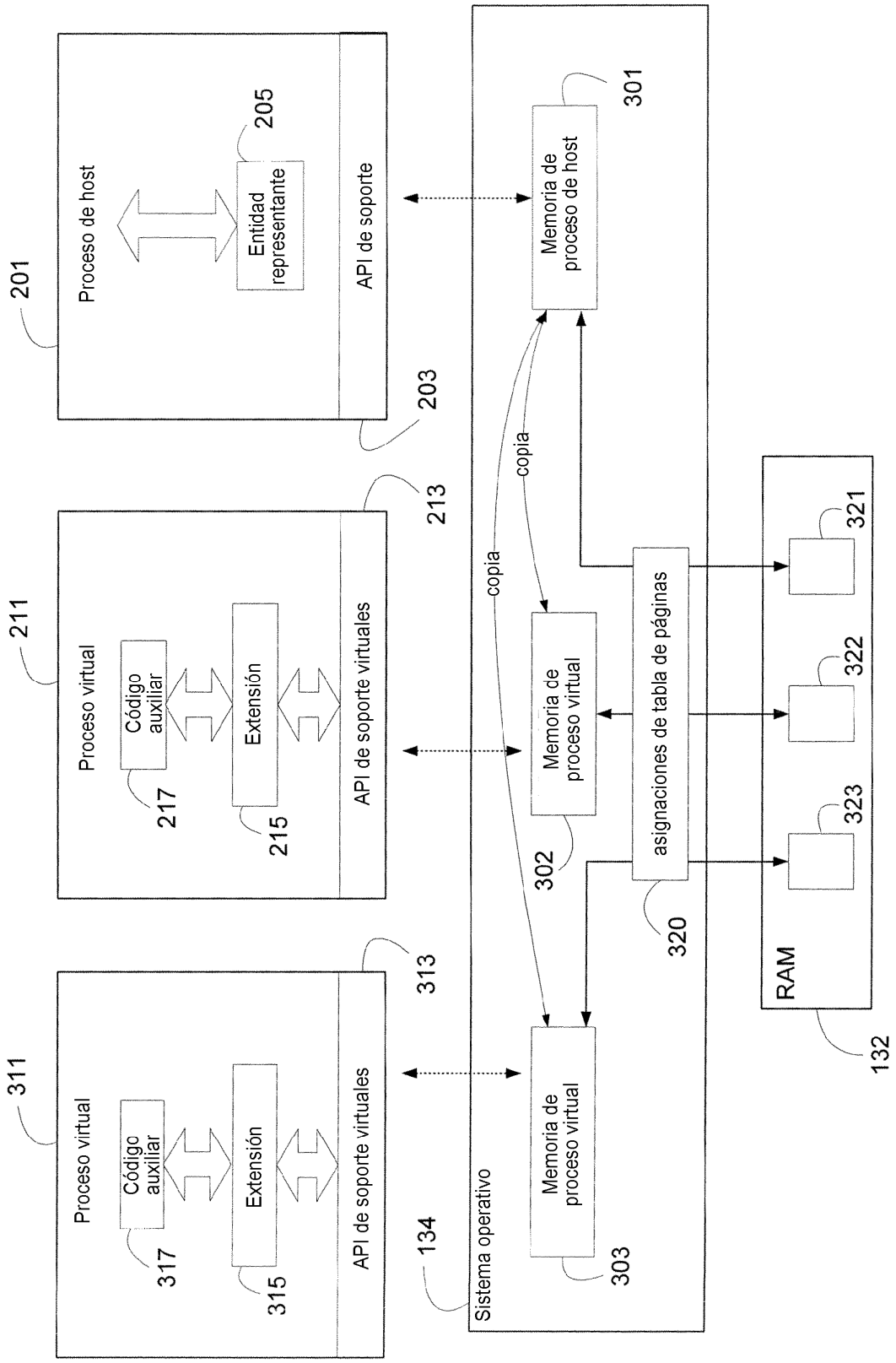
Proceso de host
Entidad representante
205
201
API de soporte
203
Proceso virtual
Código auxiliar
217
Extensión
215
211
API de soporte virtuales
213
Proceso virtual
Código auxiliar
317
Extensión
315
311
API de soporte virtuales
313
Sistema operativo
134
Memoria de proceso de host
301
Memoria de proceso virtual
302
Memoria de proceso virtual
303
copia
copia
asignaciones de tabla de páginas
320
RAM
321
322
323
132

Figura 3

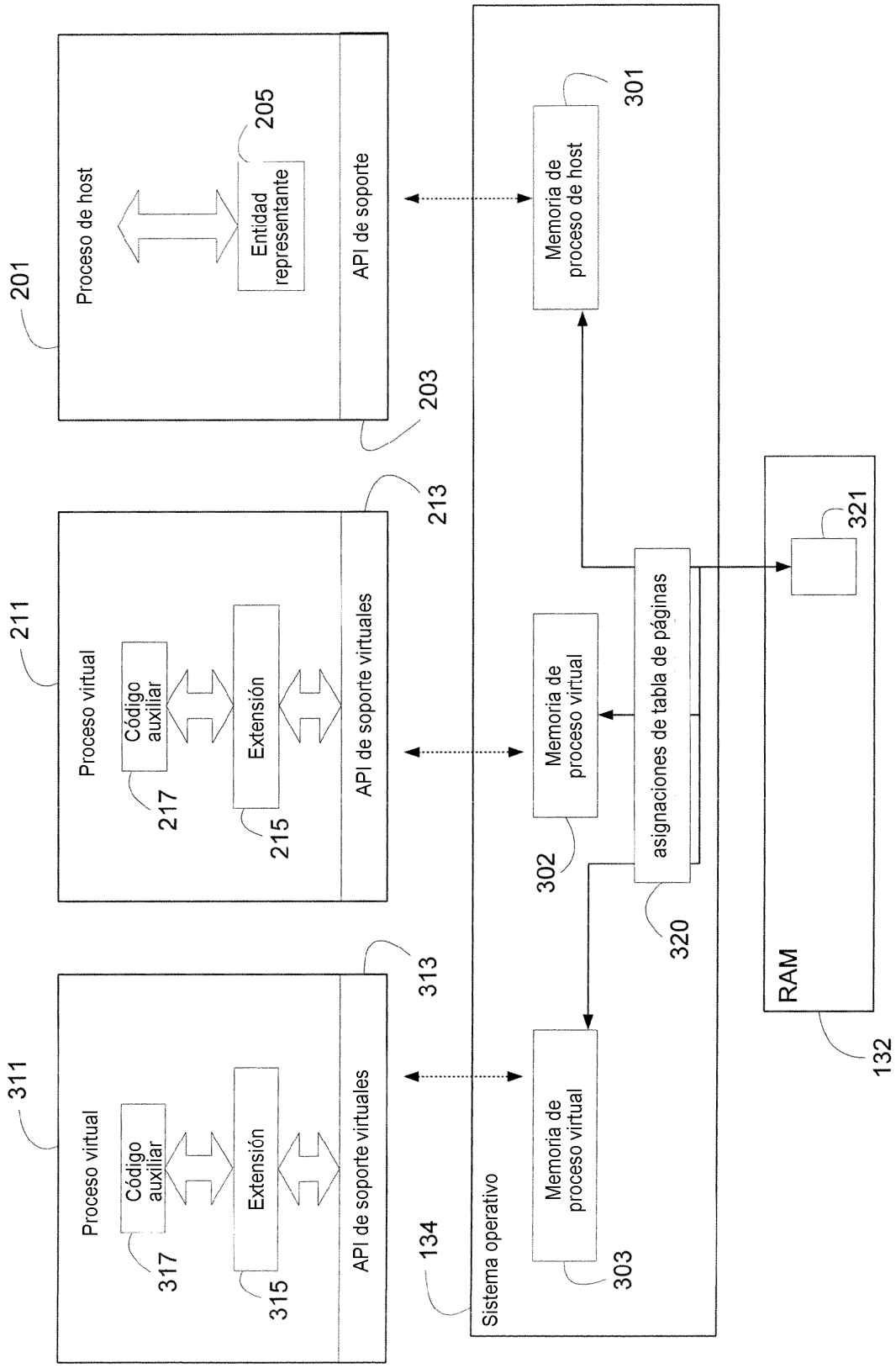
201
Proceso de host
205
Entidad representante
API de soporte
203
213
211
Proceso virtual
Código auxiliar
217
Extensión
215
API de soporte virtuales
313
311
Proceso virtual
Código auxiliar
317
Extensión
315
API de soporte virtuales
134
Sistema operativo
Memoria de proceso de host
301
Memoria de proceso virtual
302
Memoria de proceso virtual
303
asignaciones de tabla de páginas
320
RAM
321
132

Figura 4

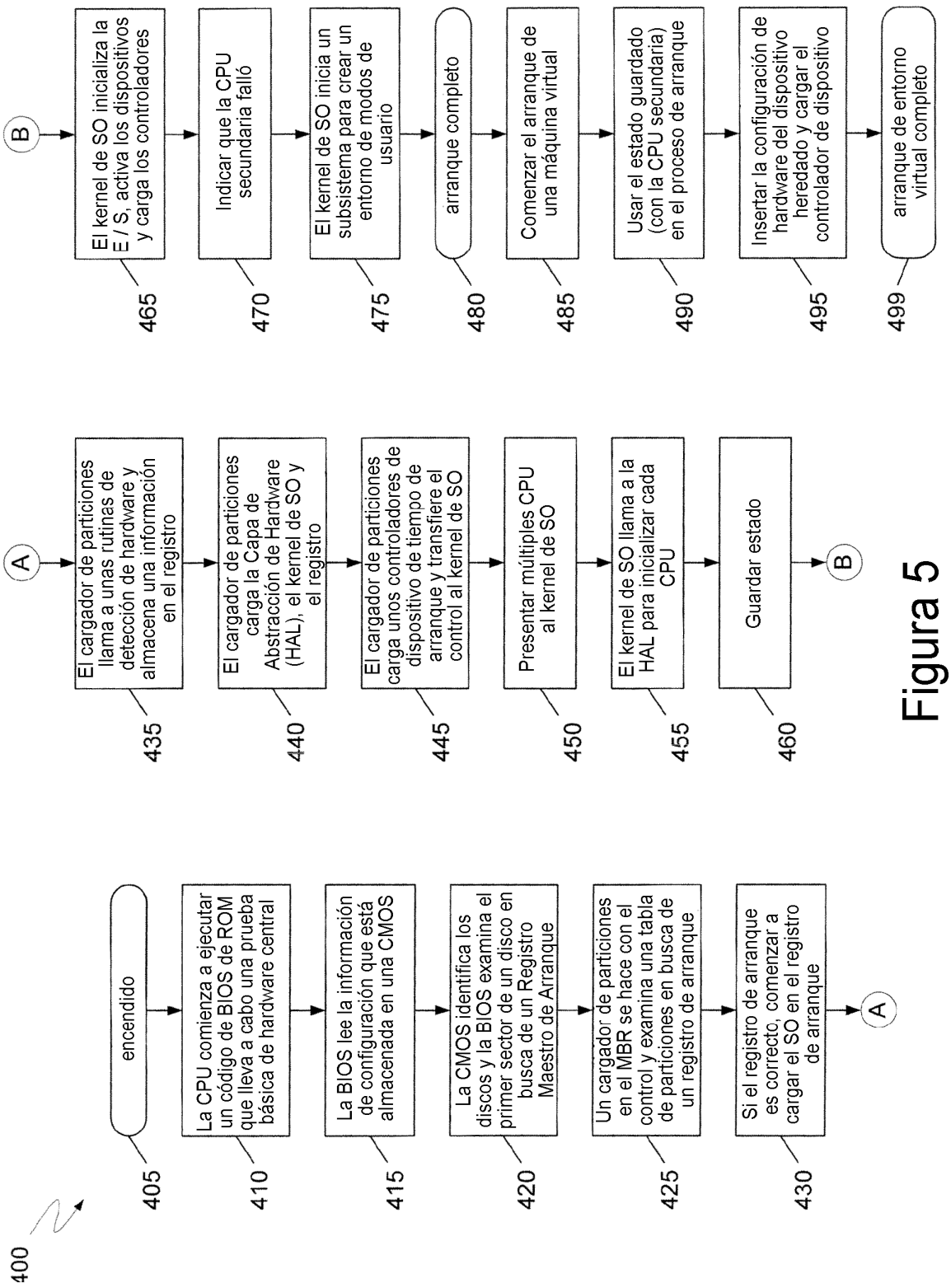
400
405
encendido
410
La CPU comienza a ejecutar un código de BIOS de ROM que lleva a cabo una prueba básica de hardware central
415
La BIOS lee la información de configuración que está almacenada en una CMOS
420
La CMOS identifica los discos y la BIOS examina el primer sector de un disco en busca de un Registro Maestro de Arranque
425
Un cargador de particiones en el MBR se hace con el control y examina una tabla de particiones en busca de un registro de arranque
430
Si el registro de arranque es correcto, comenzar a cargar el SO en el registro de arranque
A
435
El cargador de particiones llama a unas rutinas de detección de hardware y almacena una información en el registro
440
El cargador de particiones carga la Capa de Abstracción de Hardware (HAL), el kernel de SO y el registro
445
El cargador de particiones carga unos controladores de dispositivo de tiempo de arranque y transfiere el control al kernel de SO
450
Presentar múltiples CPU al kernel de SO
455
El kernel de SO llama a la HAL para inicializar cada CPU
460
Guardar estado
B
465
El kernel de SO inicializa la E / S, activa los dispositivos y carga los controladores
470
Indicar que la CPU secundaria falló
475
El kernel de SO inicia un subsistema para crear un entorno de modos de usuario
480
arranque completo
485
Comenzar el arranque de una máquina virtual
490
Usar el estado guardado (con la CPU secundaria) en el proceso de arranque
495
Insertar la configuración de hardware del dispositivo heredado y cargar el controlador de dispositivo
499
arranque de entorno virtual completo

Figura 5

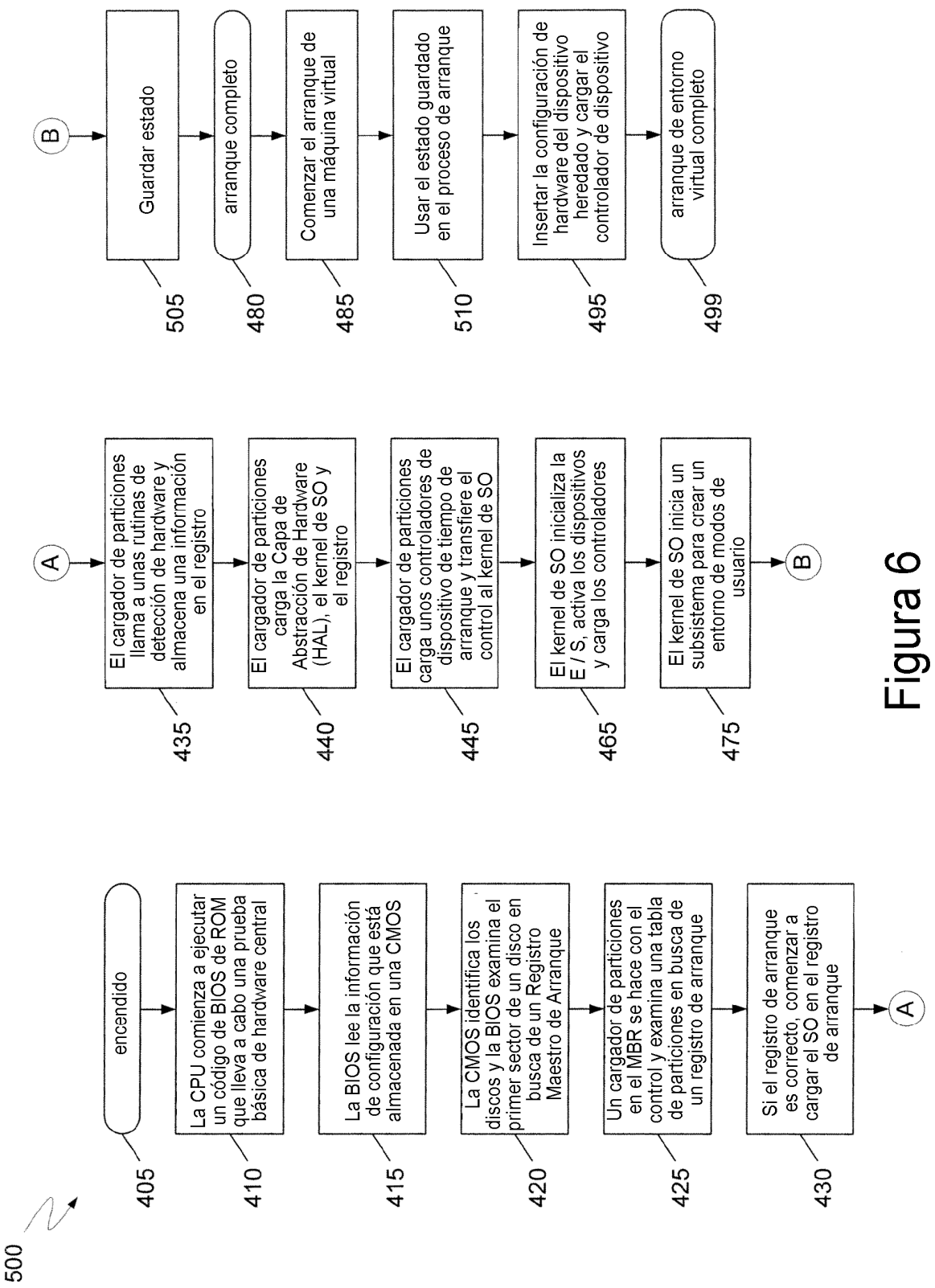
500
encendido
405
La CPU comienza a ejecutar un código de BIOS de ROM que lleva a cabo una prueba básica de hardware central
410
La BIOS lee la información de configuración que está almacenada en una CMOS
415
La CMOS identifica los discos y la BIOS examina el primer sector de un disco en busca de un Registro Maestro de Arranque
420
Un cargador de particiones en el MBR se hace con el control y examina una tabla de particiones en busca de un registro de arranque
425
Si el registro de arranque es correcto, comenzar a cargar el SO en el registro de arranque
430
A
El cargador de particiones llama a unas rutinas de detección de hardware y almacena una información en el registro
435
El cargador de particiones carga la Capa de Abstracción de Hardware (HAL), el kernel de SO y el registro
440
El cargador de particiones carga unos controladores de dispositivo de tiempo de arranque y transfiere el control al kernel de SO
445
El kernel de SO inicializa la E / S, activa los dispositivos y carga los controladores
465
El kernel de SO inicia un subsistema para crear un entorno de modos de usuario
475
B
B
Guardar estado
505
arranque completo
480
Comenzar el arranque de una máquina virtual
485
Usar el estado guardado en el proceso de arranque
510
Insertar la configuración de hardware del dispositivo heredado y cargar el controlador de dispositivo
495
arranque de entorno virtual completo
499

Figura 6

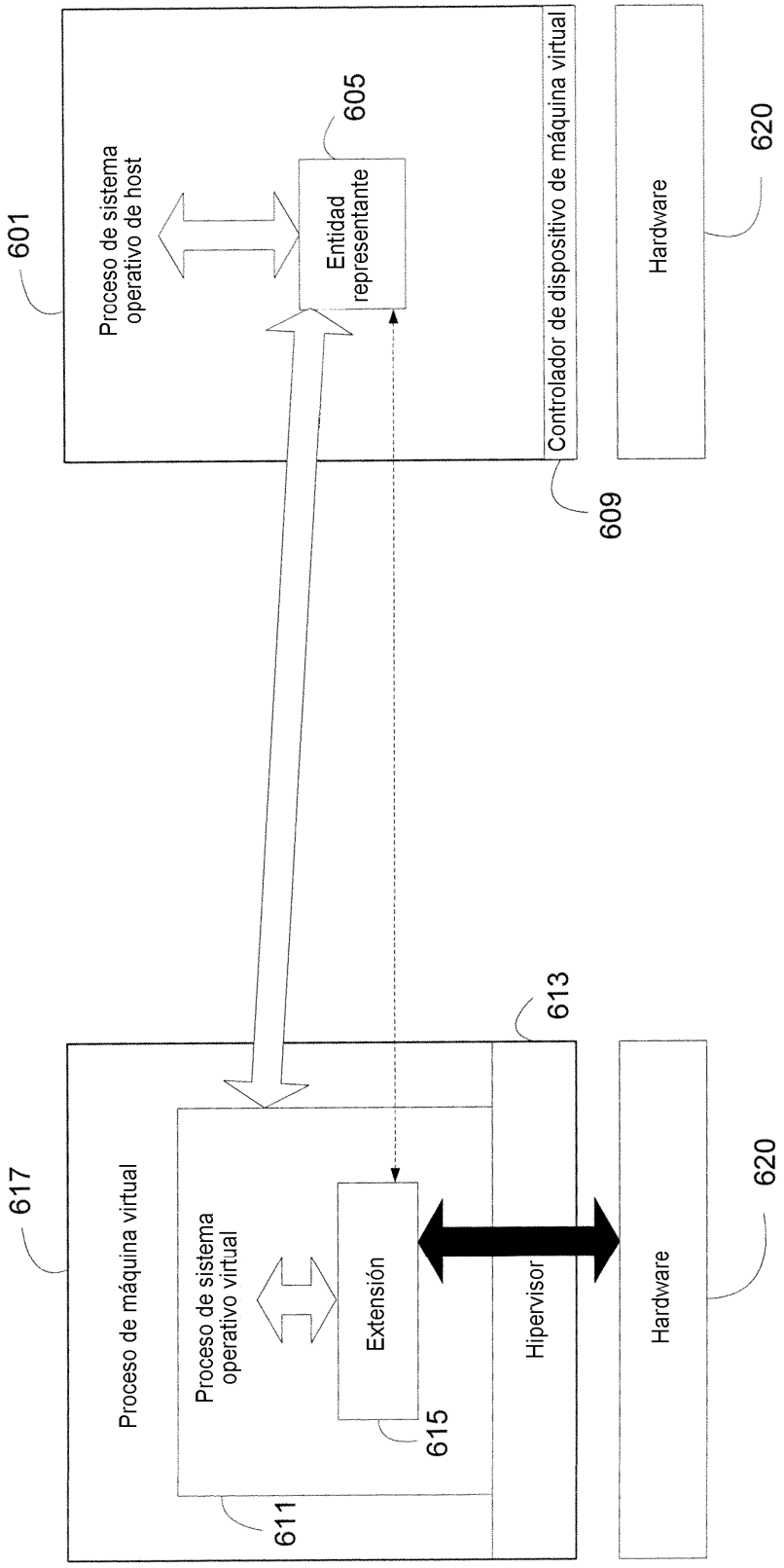
601
Proceso de sistema operativo de host
605
Entidad representante
Controlador de dispositivo de máquina virtual
609
Hardware
620
617
Proceso de máquina virtual
Proceso de sistema operativo virtual
Extensión
615
611
613
Hipervisor
Hardware
620

Figura 7