(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 776 366**

(51) Int. CI.:

G06F 8/41 (2008.01)
G06F 9/445 (2008.01)
G06F 11/07 (2006.01)
G06F 9/455 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **11.03.2015 PCT/US2015/019920**

(87) Fecha y número de publicación internacional: **17.09.2015 WO15138586**

(96) Fecha de presentación y número de la solicitud europea: **11.03.2015 E 15761173 (2)**

(97) Fecha y número de publicación de la concesión europea: **19.02.2020 EP 3117308**

(54) Título: **Sistemas y métodos para sincronización de datos y gestión de conmutación por error**

(30) Prioridad:

11.03.2014 US 201461951374 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
30.07.2020

(73) Titular/es:

IEX GROUP, INC. (100.0%)
3 World Trade Center, 58th Floor
New York, NY 10007, US

(72) Inventor/es:

CAPE, JAMES, MICHAEL;
PARK, ROBERT;
ZHANG, ALLEN;
PERKOV, ZORAN;
YU, LIETING;
SANGHVI, PRERAK, PUKHRAJ;
TATEYAMA, BEAU;
SOKOLOFF, CONSTANTINE y
QUINLAN, ERIC

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Sistemas y métodos para sincronización de datos y gestión de conmutación por error

Campo de la invención

Esta divulgación se refiere en general a entornos de tiempo de ejecución para la ejecución de aplicaciones de software y, en particular, a técnicas para mejorar el rendimiento y fiabilidad de una aplicación de software.

Antecedentes

Sistemas y programas de software, denominados en general como aplicaciones de software o aplicaciones, a menudo realizan tareas críticas en diversos campos tales como supervisión y control de sistemas médicos, transacciones financieras, supervisión y control de fabricación industrial, etc. Una aplicación que realiza una tarea crítica falla en ocasiones, por ejemplo, debido a un error de software. A través de análisis de código, pruebas y reparaciones del código, los errores de software pueden evitarse o al menos minimizarse. En ocasiones, sin embargo, la ejecución de una aplicación de software no falla debido a un error de software, sino debido a un evento en el entorno en el que se ejecuta la aplicación. Por ejemplo, puede fallar un disco u otra memoria en la que la aplicación accede a datos, puede fallar un enlace de red para acceder a uno o más componentes de la aplicación y/o datos requeridos, etc. Tales fallos de entorno pueden provocar que una aplicación de software falle.

En diversos campos, tales como los descritos anteriormente, no se espera únicamente que una aplicación de software proporcione alto rendimiento, por ejemplo, para generar resultados de cálculo requeridos tan rápido como sea posible, sino también para proporcionar estos resultados de una manera fiable. Mientras errores de software pueden evitarse o al menos minimizarse, como se ha descrito anteriormente, errores provocados por factores de entorno y/o fallos de hardware tienden a ser impredecibles. Una forma de mejorar la fiabilidad de la aplicación de software, por lo tanto, es ejecutar simultáneamente dos instancias de la aplicación de software. Incluso si una instancia falla debido a un evento de entorno, no es muy probable que en o aproximadamente en ese momento de tiempo ocurra un evento en el entorno de la otra instancia, provocando que también falle esa instancia. Por lo tanto, la otra instancia, a menudo llamada como una instancia de respaldo, puede continuar realizando las tareas de cálculo requeridas.

Este enfoque basado en redundancia presenta algunos desafíos, sin embargo. Primero, ejecutar dos instancias de la aplicación de software generalmente aumenta el coste de ejecutar la aplicación de software en términos de recursos requeridos tales como procesadores, servidores, memoria, interfaces de interconexión en red, etc. Segundo, este enfoque no es altamente escalable porque puede ocurrir otro evento de entorno que provoca un fallo en la instancia de respaldo. Mientras dos o más instancias de respaldo pueden ejecutarse simultáneamente, para aumentar la redundancia y, por tanto, la fiabilidad adicionalmente, esto también puede aumentar el coste de ejecutar la aplicación de software incluso más.

Para una ejecución eficiente de una aplicación de software, lenguajes informáticos interpretados ofrecen varios beneficios sobre sus contrapartes compiladas, tales como mayor portabilidad a través de diferentes arquitecturas informáticas. Ejecutar código en un intérprete puede ser dramáticamente más lento, sin embargo, que ejecutar el mismo código compilado en el lenguaje de máquina nativo de un procesador. Para superar esta desventaja, muchos intérpretes pueden mejorar el rendimiento incluyendo compilación justo a tiempo (JIT), en la que al menos una porción de los códigos de bytes del intérprete se compila al código nativo del procesador. El intérprete puede ejecutar a continuación el código nativo en lugar de interpretar el código de bytes para esa porción del programa. Esta característica se proporciona en varios tiempos de ejecución de intérprete, incluyendo la implementación de HotSpot estándar de la máquina virtual Java (JVM) y Tiempo de Ejecución de Lenguaje Común de Microsoft (comúnmente conocido como .NET), y puede ofrecer un rendimiento comparable a lenguajes compilados antes de tiempo (AIT) tradicionales.

Otra mejora de rendimiento es posible con lenguajes compilados JIT, en los que a medida que el intérprete ejecuta un código de bytes particular (u otra representación intermedia, o código fuente), el intérprete puede recopilar información de perfil acerca del código ejecutado o al menos una parte del mismo, que habilita un mayor grado de optimización cuando se compila el código o una porción del mismo. Esta optimización adaptativa se proporciona en el HotSpot JVM y puede ofrecer en ocasiones mejor rendimiento que código compilado antes de tiempo. Un principal inconveniente de esta técnica es que el código o una porción del mismo debe ejecutarse muchas veces (por ejemplo, docenas, cientos, miles o incluso más veces) usando al intérprete para recopilar suficiente información de perfil para optimizar la compilación. Durante la fase de perfilado, la aplicación de software habitualmente se ejecuta más lenta que una versión compilada compilada usando una técnica de compilación tradicional tal como compilación AIT. Únicamente después de que se complete el perfilado del código, el JIT adaptativo puede ofrecer rendimiento mejorado.

El documento US 2011/0088021 divulga un sistema para configurar opciones de compilador para optimizar el rendimiento de ejecución de código. Diferentes instancias de código ejecutable se generan usando diferentes

opciones de optimización de compilador en el mismo código que se ejecutan a continuación en paralelo en múltiples núcleos. Los resultados de la instancia que primero completa la ejecución entre las ejecuciones paralelas se proporcionan como resultados para el código de ejecución.

El documento US 8621275 divulga un sistema para replicar la ejecución de una instancia de aplicación primaria a uno o más anfitriones de respaldo. Cuando la instancia de aplicación primaria falla, una instancia de respaldo se asciende a primaria y continúa la ejecución.

Sumario

La invención se define mediante las reivindicaciones independientes adjuntas. Las reivindicaciones dependientes definen realizaciones preferidas. Realizaciones de sistemas y métodos descritos en este documento proporcionan Sincronización de Datos y Gestión de Conmutación por Error (DSFM), a través de las cuales puede aumentarse la fiabilidad de una aplicación de software facilitando ejecución simultánea de varias instancias no idénticas de la aplicación de software. Mientras esto puede aumentar el coste de ejecución de la aplicación de software, a diferencia de técnicas basadas meramente en redundancia, las instancias no idénticas pueden proporcionar una mejora de rendimiento. Específicamente, cada instancia puede generarse para optimizar un respectivo objetivo diferente tal como maximizar, minimizar o utilizar, de acuerdo con un límite especificado, uno o más recursos tales como memoria, interfaces de red, capacidad de procesamiento, número de procesadores disponibles, etc. Algunas instancias pueden optimizarse para ejecución usando información de tiempo de ejecución disponible de ejecuciones anteriores y/o usando compilación justo a tiempo (JIT). Tales instancias pueden generarse compilando el código fuente de la aplicación de software usando diferentes compiladores y/o usando diferentes opciones de compilador que pueden optimizar uno o más objetivos especificados, tales como los descritos anteriormente.

Ya que cada instancia se deriva a partir de sustancialmente el mismo código fuente, cada instancia produce, en el fondo, los mismos resultados que cualquier otra instancia, es decir, cada instancia es probable que produzca los mismos resultados que se espera que produzca la aplicación de software. Diferentes instancias pueden optimizarse de forma diferente, sin embargo, y, por lo tanto, pueden emitir uno o más resultados en una secuencia de resultados en momentos diferentes durante su respectiva ejecución, incluso aunque la ejecución de las diversas instancias de la aplicación de software se inicie en o aproximadamente en el mismo momento. Si se espera que la aplicación de software produzca una secuencia de resultados, para cada resultado, una implementación de un sistema de DSFM puede designar/etiquetar una correspondiente salida de una instancia que produjo la salida antes de todas las demás instancias como el resultado requerido y puede descartar las salidas de otras instancias como duplicadas. En general, diferentes instancias pueden emitir primero diferentes resultados. Designando/etiquetando las respectivas salidas producidas más temprano como los resultados sucesivos de la aplicación de software, una implementación del sistema de DSFM puede mejorar el rendimiento global de la aplicación de software.

Independientemente de si una implementación de sistema de DSFM designa/etiqueta o selecciona una salida particular de una instancia particular como un resultado particular de la aplicación de software, cada instancia realiza todos los cálculos que se requieren para generar ese resultado. Estas instancias pueden estar en diferentes estados porque algunas instancias, debido a respectivas optimizaciones diferentes de las mismas, pueden haber realizado cálculos adicionales que no se requieren que produzcan el resultado particular. Sin embargo, cuando todas las instancias han completado los cálculos que se requieren que produzcan un resultado particular, una implementación de sistema de DSFM considera que estas instancias están en sincronización o en *sinc*.

Una implementación de sistema de DSFM puede designar/etiquetar una de estas instancias como una instancia primaria. Opcionalmente, una implementación de sistema de DSFM puede designar/etiquetar las salidas de la instancia primaria como los resultados de la aplicación de software, y puede descartar las salidas de la una o más otras instancias, designadas/etiquetadas como instancia o instancias secundarias. Si la instancia primaria falla después de que se produzcan k resultados, colectivamente por todas las instancias, como se ha descrito anteriormente, o por la instancia primaria sola, una implementación de sistema de DSFM puede designar/etiquetar una de las instancias secundarias como una nueva instancia primaria porque, como se ha descrito anteriormente, todas estas instancias están en sinc, aunque no en un estado idéntico, después de que se produzcan los k resultados. Los (k+1)-ésimo y uno o más resultados posteriores pueden producirse por la instancia secundaria redesignada/reetiquetada como la nueva instancia primaria.

Si la recientemente designada/etiquetada instancia primaria falla después de, por ejemplo, el cálculo de n resultados, aún otra instancia secundaria estaría en sinc, y una implementación de sistema de DSFM puede redesignar/reetiquetar esa otra instancia secundaria como la nueva instancia primaria. Mientras el coste de ejecutar una aplicación de software aumentaría generalmente con el número de instancias, las diferentes instancias, debido a diferentes respectivas optimizaciones de las mismas, puede tener diferentes rendimientos asociados con la producción de diferentes resultados, y la selección de la salida más temprana puede aumentar el rendimiento global de la aplicación de software. Incluso aunque las diversas instancias son no idénticas, varias implementaciones de un sistema de DSFM puede determinar los instantes en los que las diversas instancias están en sinc, y pueden habilitar, por lo tanto, la conmutación por error desde una instancia a otra instancia diferente, probablemente en un estado diferente que el de la instancia fallida, sin afectar a los resultados producidos, y pueden aumentar, por lo tanto, la

fiabilidad de la aplicación de software.

Por consiguiente, en un aspecto se proporciona un método para ejecutar una aplicación de software de forma expeditiva en al menos un procesador informático. El método incluye ejecutar simultáneamente varias instancias de la aplicación de software en uno o más procesadores informáticos, en el que cada instancia se compila de acuerdo con una respectiva opción de compilador que es diferente de respectivas opciones de compilador usadas para compilar todas las demás instancias a partir de las varias instancias. Para cada uno de un primer conjunto de resultados a producir por la aplicación de software, el método incluye supervisar, correspondiendo al resultado, una respectiva salida generada por cada instancia. Para cada uno del primer conjunto de resultados a producir por la aplicación, el método también incluye etiquetar, de las salidas supervisadas, la salida que ocurre más temprano como la salida de la aplicación que corresponde al resultado, y etiquetar todas las demás salidas como duplicadas. De esta manera, cada uno del primer conjunto de resultados se obtiene a partir de una instancia que produjo ese resultado particular antes de todas las demás instancias, acelerando de este modo un rendimiento informático del uno o más procesadores informáticos en la ejecución de la aplicación de software.

En algunas realizaciones, las varias instancias incluyen una primera instancia y una segunda instancia, y la opción de compilador para la primera instancia incluye compilación antes de tiempo (AIT). Por ejemplo, la primera instancia puede compilarse usando compilación AIT. La opción de compilador para la segunda instancia puede incluir compilación justo a tiempo (JIT). Compilación JIT de la segunda instancia puede basarse en, al menos en parte, información de tiempo de ejecución obtenido a partir de una o más ejecuciones anteriores de la segunda instancia. Al menos una porción de código fuente de la aplicación de software puede especificarse usando un lenguaje de programación que se interpreta al menos parcialmente.

En algunas realizaciones, las varias instancias incluyen una primera instancia, y la opción de compilador para la primera instancia puede ser uno o más de uso de memoria sin restricciones, minimización de uso de memoria, maximización de operaciones concurrentes y concurrencia restringida de operaciones. Por lo tanto, en un ejemplo, la primera instancia puede maximizar el cálculo paralelo aprovechando el uso de memoria sin restricciones. En otro ejemplo, la primera instancia puede minimizar el uso de memoria, etc.

En algunas realizaciones, el método incluye adicionalmente etiquetar una instancia de las varias instancias como una instancia primaria, y etiquetar todas las demás instancias como instancias secundarias. El método puede incluir también, para cada resultado de una segunda pluralidad de resultados a producir por la aplicación, suprimir, correspondiendo al resultado, respectivas salidas de las instancias secundarias. El método puede incluir también identificar un fallo de la instancia primaria después de producir k resultados a partir de la segunda pluralidad de resultados, donde $k \geq 1$. La instancia primaria puede etiquetarse a continuación como una instancia fallida. El método puede incluir adicionalmente seleccionar una instancia secundaria que ejecutó lógica asociada con cálculo de cada uno de los k resultados, y reetiquetar las instancias secundarias seleccionadas como la instancia primaria, con lo que la instancia primaria reetiquetada produce (k+1)-ésimo resultado. Debido a que la instancia secundaria seleccionada ejecutó lógica asociada con el cálculo de cada uno de los k resultados, la instancia secundaria seleccionada está probablemente en sinc con la instancia primaria antes del fallo de la misma. Por lo tanto, la ejecución de la aplicación a través de dos o más instancias puede conmutar por error desde una instancia a otra, generalmente sin introducir retardos (tal como los asociados con el reinicio de una aplicación) y/o errores sustanciales.

En otro aspecto, se proporciona un método para habilitar conmutación por error para una aplicación de software. El método incluye supervisar por un controlador basado en procesador ejecuciones simultáneas de una instancia primaria de la aplicación de software y una primera instancia secundaria de la aplicación. La instancia primaria se compila de acuerdo con una primera opción de compilador, y la primera instancia secundaria se compila de acuerdo con una opción de compilador que es diferente de la primera opción de compilador. El método también incluye detectar un fallo de la instancia primaria después de la producción de k resultados de la aplicación de software, donde $k \geq 1$. Además, el método incluye confirmar que la primera instancia secundaria ha calculado operaciones requeridas para calcular el k-ésimo resultado, y etiquetar la primera instancia secundaria como la instancia primaria. Debido a que la primera instancia secundaria calculó las operaciones requeridas para calcular cada uno de los k resultados, la primera instancia secundaria está probablemente en sinc con la instancia primaria antes del fallo de la misma. Por lo tanto, la ejecución de la aplicación a través de dos o más instancias puede conmutar por error desde una instancia a otra, generalmente sin introducir retardos (tal como los asociados con el reinicio de una aplicación) y/o errores sustanciales.

El controlador puede suprimir salidas de la primera instancia secundaria que corresponden a los k resultados, es decir, antes del fallo de la instancia etiquetada como la instancia primaria. La primera opción de compilador puede incluir compilación justo a tiempo (JIT), y la opción de compilador usada para la primera instancia secundaria puede incluir compilación antes de tiempo (AIT). En algunas realizaciones, el método incluye adicionalmente supervisar, por el controlador, ejecución simultánea de una segunda instancia secundaria de la aplicación. La segunda instancia secundaria se compila de acuerdo con una opción de compilador que es diferente tanto de la primera opción de compilador como de la opción de compilador usada para compilar la primera instancia secundaria. Si el método no confirma que la primera instancia secundaria ha calculado las operaciones requeridas para calcular el k-ésimo

resultado, el método puede incluir confirmar que la segunda instancia secundaria tiene las operaciones calculadas requeridas para calcular el k-ésimo resultado. La segunda instancia secundaria puede etiquetarse a continuación como la instancia primaria, permitiendo conmutación por error eficiente y sin errores desde la instancia anteriormente etiquetada como la instancia primaria a la segunda instancia secundaria, que se reetiqueta como la instancia primaria.

En otro aspecto, un sistema para ejecutar una aplicación de software de forma expeditiva en al menos un procesador informático incluye un primer procesador y una primera memoria en comunicación eléctrica con el primer procesador. La primera memoria incluye instrucciones que, cuando se ejecutan por una unidad de procesamiento que puede incluir el primer procesador y/o un segundo procesador, programan la unidad de procesamiento para ejecutar simultáneamente varias instancias de la aplicación de software en uno o más procesadores informáticos. Cada instancia se compila de acuerdo con una respectiva opción de compilador que es diferente de respectivas opciones de compilador usadas para compilar todas las demás instancias a partir de las varias instancias. Para cada uno de un primer conjunto de resultados a producir por la aplicación, las instrucciones programan la unidad de procesamiento para supervisar, correspondiendo al resultado, una respectiva salida generada por cada instancia. Para cada uno del primer conjunto de resultados a producir por la aplicación, las instrucciones también programan la unidad de procesamiento para etiquetar, de las salidas supervisadas, la salida que ocurre más temprano como la salida de la aplicación que corresponde al resultado, y para etiquetar todas las demás salidas como duplicadas. De esta manera, cada uno del primer conjunto de resultados se obtiene a partir de una instancia que produjo ese resultado particular antes de todas las demás instancias, acelerando de este modo un rendimiento informático del uno o más procesadores informáticos en la ejecución de la aplicación de software. En diversas realizaciones, las instrucciones pueden programar la unidad de procesamiento para realizar una o más de las etapas de método descritas anteriormente.

En otro aspecto, un sistema para habilitar conmutación por error para una aplicación de software incluye un primer procesador y una primera memoria en comunicación eléctrica con el primer procesador. La primera memoria incluye instrucciones que, cuando se ejecutan por una unidad de procesamiento que puede incluir el primer procesador y/o un segundo procesador, programan la unidad de procesamiento para supervisar ejecuciones simultáneas de una instancia primaria de la aplicación de software y una primera instancia secundaria de la aplicación. La instancia primaria se compila de acuerdo con una primera opción de compilador, y la primera instancia secundaria se compila de acuerdo con una opción de compilador que es diferente de la primera opción de compilador. Las instrucciones también programan la unidad de procesamiento para detectar un fallo de la instancia primaria después de la producción de k resultados de la aplicación de software, donde $k \geq 1$. Además, las instrucciones programan la unidad de procesamiento para confirmar que la primera instancia secundaria ha calculado operaciones requeridas para calcular el k-ésimo resultado, y para etiquetar la primera instancia secundaria como la instancia primaria. Como tal, la ejecución de la aplicación a través de dos o más instancias puede conmutar por error desde una instancia a otra, generalmente sin introducir retardos (tal como los asociados con el reinicio de una aplicación) y/o errores sustanciales. En diversas realizaciones, las instrucciones pueden programar la unidad de procesamiento para realizar una o más de las etapas de método descritas anteriormente.

En otro aspecto, un artículo de fabricación que incluye un medio de almacenamiento no transitorio ha almacenado en el mismo instrucciones que, cuando se ejecutan por una unidad de procesamiento, programan la unidad de procesamiento, que está en comunicación electrónica con una memoria, para ejecutar simultáneamente varias instancias de la aplicación de software en uno o más procesadores informáticos. Cada instancia se compila de acuerdo con una respectiva opción de compilador que es diferente de respectivas opciones de compilador usadas para compilar todas las demás instancias a partir de las varias instancias. Para cada uno de un primer conjunto de resultados a producir por la aplicación, las instrucciones programan la unidad de procesamiento para supervisar, correspondiendo al resultado, una respectiva salida generada por cada instancia. Para cada uno del primer conjunto de resultados a producir por la aplicación, las instrucciones también programan la unidad de procesamiento para etiquetar, de las salidas supervisadas, la salida que ocurre más temprano como la salida de la aplicación que corresponde al resultado, y para etiquetar todas las demás salidas como duplicadas. En diversas realizaciones, las instrucciones pueden programar la unidad de procesamiento para realizar una o más de las etapas de método descritas anteriormente.

En otro aspecto, un artículo de fabricación que incluye un medio de almacenamiento no transitorio ha almacenado en el mismo instrucciones que, cuando se ejecutan por una unidad de procesamiento, programan la unidad de procesamiento, que está en comunicación electrónica con una memoria, para supervisar ejecuciones simultáneas de una instancia primaria de la aplicación de software y una primera instancia secundaria de la aplicación. La instancia primaria se compila de acuerdo con una primera opción de compilador, y la primera instancia secundaria se compila de acuerdo con una opción de compilador que es diferente de la primera opción de compilador. Las instrucciones también programan la unidad de procesamiento para detectar un fallo de la instancia primaria después de la producción de k resultados de la aplicación de software, donde $k \geq 1$. Además, las instrucciones programan la unidad de procesamiento para confirmar que la primera instancia secundaria ha calculado operaciones requeridas para calcular el k-ésimo resultado, y para etiquetar la primera instancia secundaria como la instancia primaria. En diversas realizaciones, las instrucciones pueden programar la unidad de procesamiento para realizar una o más de las etapas de método descritas anteriormente.

Un efecto técnico de diversas realizaciones de la presente invención es que el rendimiento global de una aplicación de software que se espera que produzca una secuencia de resultados puede aumentarse seleccionando para cada resultado la salida más temprana de varias instancias que producen simultáneamente salidas que corresponden a ese resultado. Los momentos en los que las diferentes instancias pueden producir salidas que corresponden a un resultado particular pueden ser diferentes debido a diferencias en la manera en la que las diferentes instancias se compilan. Algunas instancias pueden producir inicialmente salidas más lentas que algunas otras instancias, pero pueden producir posteriormente salidas más rápidas que esas instancias. Seleccionando la salida más temprana para cada resultado, de forma efectiva, la aplicación de software se ejecuta a través de las instancias que fueron las más rápidas en producir salidas que corresponden respectivamente a diferentes resultados, acelerando de este modo ejecución de la aplicación de software en general.

Otro efecto técnico de diversas realizaciones de la presente invención es que mientras se aumenta el rendimiento como se ha descrito anteriormente, las diferentes instancias pueden mantenerse en sincronización, aunque no en el mismo estado exacto, en el cálculo de cada uno de una serie de resultados a producir por la aplicación de software. Por lo tanto, si una instancia falla después de la producción de uno o más resultados, puede usarse otra instancia, probablemente la más rápida en producir el siguiente resultado entre las que no han fallado, para producir el siguiente resultado. Por lo tanto, puede conseguirse una conmutación por error segura, es decir, generalmente sin errores, sin tener que incurrir en excesivos retardos, tales como los asociados con el reinicio de la aplicación, mientras se mejora el rendimiento global de la aplicación de software.

Estos y otros objetivos, junto con ventajas y características de las realizaciones divulgadas en este documento, serán más evidentes a través de referencia a la siguiente descripción, los dibujos adjuntos y las reivindicaciones. Adicionalmente, debe apreciarse que las características de las diversas realizaciones descritas en este documento no son mutuamente excluyentes y pueden existir en diversas combinaciones y permutaciones.

Breve descripción de los dibujos

En los dibujos, caracteres de referencia similares generalmente se refieren a las mismas partes a lo largo de todas las diferentes vistas. También, los dibujos no están necesariamente a escala, haciendo énfasis en su lugar generalmente en la ilustración de los principios de la invención. En la siguiente descripción, diversas realizaciones de la presente invención se describen con referencia a los siguientes dibujos, en los que:

La Figura 1 ilustra los usos de diversas realizaciones de un sistema de DSFM;

La Figura 2 representa un diagrama de gráfico de datos que ilustra conmutación por error desde una instancia primaria de una aplicación de software a una instancia secundaria, usando una realización de un sistema de DSFM;

La Figura 3 representa un diagrama de flujo lógico que ilustra una transición desde una instancia de una aplicación de software a otra instancia, usando una realización de un sistema de DSFM;

Las Figuras 4 y 5 ilustran esquemáticamente una mejora de rendimiento de una aplicación de software usando una realización de un sistema de DSFM; y

La Figura 6 representa esquemáticamente un controlador de DSFM, de acuerdo con una realización.

Descripción detallada

Con referencia a la Figura 1, un usuario 101 puede desear ejecutar una aplicación de software particular. Un sistema de Sincronización de Datos y Gestión de Conmutación por Error (DSFM) 102 puede ejecutar dos instancias de la aplicación de software, una designada/etiquetada como una instancia primaria y otra designada/etiquetada como una instancia secundaria. La instancia primaria puede fallar, por ejemplo, debido a fallo de hardware, error de interconexión de red, etc., y la ejecución puede transferirse a la instancia secundaria (también llamada una instancia de respaldo). En algunas realizaciones, la instancia secundaria puede no estar en el mismo estado exacto que la primaria, por ejemplo, porque incluso aunque las instancias primaria y secundaria se derivan a partir de código fuente idéntico, las dos instancias pueden compilarse de forma diferente. Por lo tanto, la instancia secundaria puede no seguir la misma ruta de código que la de la instancia primaria. Como tal, la ejecución de la instancia secundaria puede ser diferente en comparación con la de la instancia primaria, resultando en, por ejemplo, resultados diferentes y/o potencialmente deficientes, por ejemplo, en términos de velocidad de ejecución.

Por ejemplo, la primera y segunda instancias, incluso si estuvieran en estados idénticos en el inicio, pueden distanciarse durante la ejecución, e incluso cuando están en sinc, es decir, implementando generalmente la misma lógica general, las dos instancias pueden no estar exactamente en el mismo estado. Por ejemplo, en algunos sistemas de software que se codifican en lenguajes interpretados o parcialmente interpretados tales como, pero sin limitación, Java, una compilación justo a tiempo (JIT) puede traducir un código de código de bytes en un código

máquina, mientras busca una ruta de código óptima a seguir durante la ejecución. Si se usa compilación JIT para generar o bien la instancia primaria o bien la secundaria, pero no ambas, en un momento particular, la primera y segunda instancias pueden no ejecutar exactamente las mismas instrucciones. Como un ejemplo, considérese un segmento de código:

```
If (eval(condition) == true)
       x=fl(A);
else
       x=f2(A);
output (x);
```

Una instancia (indicada como instancia A) de este segmento de código, compilado usando un compilador, puede evaluar primero la condición, a continuación determinar si la condición es verdadera, y puede calcular a continuación f1 o f2 en consecuencia, para producir el resultado x.

El mismo u otro compilador puede tener conocimiento de que la evaluación de la condición normalmente tarda mucho tiempo (por ejemplo, unos pocos milisegundos), y la evaluación de f1 y f2 también tarda algún tiempo (por ejemplo, unos pocos milisegundos), pero que los tres cálculos no dependen entre sí. Por lo tanto, otra instancia (indicada como instancia B) del mismo segmento de código, compilado usando el conocimiento acerca de los tiempos de ejecución, usando el otro compilador o una opción diferente del mismo compilador, puede calcular tanto f1 y f2 mientras la evaluación de la condición está en marcha. Después de que se completa la evaluación, la instancia B puede elegir fácilmente entre los resultados de f1 y f2, produciendo de este modo la salida más rápido de lo que lo haría la instancia A. Por lo tanto, en un momento particular de tiempo, la instancia A del segmento de código puede estar evaluando la condición únicamente, mientras que la instancia B también puede estar calculando adicionalmente f1 y f2.

En otro ejemplo más, un compilador usado para generar otra instancia (indicada como instancia C) del segmento de código puede tener conocimiento de que la condición es falsa más a menudo que lo que no lo es. Por lo tanto, la instancia C puede calcular únicamente f2 y no f1, mientras la evaluación de la condición está en marcha, y calcular f1 únicamente si la condición se determina que es verdadera. Por lo tanto, el cálculo de x puede acelerarse en relación con el de la instancia A, que se compiló sin usar ningún conocimiento adicional, en una mayoría de situaciones pero no en todas. En un momento particular de tiempo la instancia A puede estar evaluando la condición únicamente, mientras la instancia C puede estar evaluando tanto la condición como f2. Por lo tanto, incluso aunque, las instancias A, B y C generalmente implementan la misma lógica que se especifica por el segmento de código, en un momento particular de tiempo, cada una de estas instancias pueden ejecutar diferentes instrucciones.

Si dos o más de tales instancias compiladas usando diferentes técnicas de compilación, diferentes compiladores y/o diferentes opciones de compilador se ejecutan simultáneamente, la una o más instancias que se retardan en general en la producción de resultados de la aplicación en relación con otra instancia, pueden configurarse para suprimir sus respectivas salidas, por ejemplo, para evitar la duplicación de resultados. Para ilustrar, durante la ejecución, la instancia primaria (por ejemplo, la instancia B) puede haber generado una salida particular (por ejemplo, resultado x) y, por lo tanto, la instancia secundaria (por ejemplo, instancias A o C) puede no producir un duplicado de esa salida, incluso cuando las instancias A y C realizan los cálculos requeridos para producir esa salida. En algunas realizaciones, la instancia designada/etiquetada para producir salidas de la misma como resultados de la aplicación puede llamarse una instancia primaria (por ejemplo, la instancia B), y la una o más otras instancias (por ejemplo, la instancia A y la instancia C) pueden llamarse instancia o instancias secundarias. Habiendo realizado la instancia o instancias secundarias las operaciones asociadas con cada resultado a producir por la aplicación y suprimiendo únicamente las salidas de la instancia o instancias secundarias, la instancia o instancias secundarias se mantienen en sinc con la instancia primaria y en un estado tan cercano al estado de la primaria como sea posible.

En algunas implementaciones, la compilación JIT para la primera y segunda instancias pueden generar diferentes rutas de código con diferentes instrucciones para la primera y segunda instancias, como se ha descrito anteriormente usando el ejemplo segmento de código. En algunas implementaciones, la primera y segunda instancias pueden tener diferentes rutas de código porque se usan diferentes clases de compiladores (por ejemplo, JIT y antes de tiempo (AIT)) para compilar la primera y segunda instancias. En algunas implementaciones, puede usarse el mismo compilador para compilar tanto la instancia primaria como la instancia secundaria, pero pueden usarse diferentes opciones de compilador durante respectivas compilaciones. Ejemplos de opciones de compilador incluyen, pero sin limitación, minimizar el uso de memoria, uso de memoria sin restricciones, maximizar operaciones concurrentes y/o parcialmente solapantes (por ejemplo, usando procesadores paralelos, varios hilos, etc.), limitar el número de operaciones concurrentes y/o parcialmente solapantes a un límite especificado, etc. En algunas implementaciones, se detecta que una instancia primaria ha fallado y la ejecución se conmuta a una instancia secundaria. La ejecución de instancia secundaria puede seguir una ruta de código diferente y, por lo tanto, el rendimiento y eficiencia de la instancia secundaria puede no coincidir con la de la instancia primaria, por ejemplo, 104.

Diversas realizaciones de un sistema de DSFM descrito en este documento pueden aprovecharse de una opción en

algunos tiempos de ejecución JIT usando que todo el programa puede compilarse antes de tiempo. Por ejemplo, en el HotSpot JVM, la opción de línea de comando -Xcomp puede provocar que el tiempo de ejecución compile cada método ejecutado inmediatamente en lugar de someter el método a las relativamente más lentas fases de interpretación y perfilado. En .NET, la utilidad NGen puede compilar programas de código de bytes a código nativo en el momento de instalación en lugar de hacerlo cada vez que se ejecuta el programa. Mientras estas opciones pueden diferir en su operación, generalmente ambas tienen el mismo efecto de transformar JIT en compilación antes de tiempo.

En algunas realizaciones, se ejecutan simultáneamente dos instancias de un único programa de código de bytes. Simultáneamente generalmente significa con al menos un solapamiento parcial en sus respectivos momentos de ejecución. La primera instancia usa la compilación antes de tiempo opcional mientras que la segunda se ejecuta por el tiempo de ejecución JIT adaptativo estándar. Se suministra a ambas instancias con entradas idénticas y pueden producir idénticas salidas, pero a diferentes velocidades. En general, inicialmente la primera instancia es significativamente más rápida ya que usa código compilado, mientras que la segunda instancia continúa perfilando el código en el intérprete. Ya que la segunda instancia compila gradualmente el programa, su rendimiento habitualmente aumenta y finalmente puede superar el rendimiento de la primera instancia, en parte debido a la mayor optimización disponible con el compilador JIT adaptativo.

Las diferentes ejecuciones de las dos instancias pueden representarse como una única ejecución de una aplicación de software. Para este fin, las respectivas salidas de las dos instancias pueden fusionarse a través de un proceso de filtro. En algunas realizaciones, el proceso de filtro recopila los mensajes de salida de las dos instancias, pero únicamente emite un mensaje de salida en respuesta al primero o al producido más temprano de los mensajes fusionados, es decir, un mensaje producido por una instancia más rápida y se descartan los mensajes duplicados producidos por la otra instancia relativamente más lenta. Ya que cada instancia se espera que produzca los mismos mensajes de salida en el mismo orden, el filtro puede emitir de forma efectiva mensajes desde la más rápida de las dos instancias. En efecto, la combinación de las dos instancias y el filtro opera como una única instancia tanto con el rendimiento inicial más rápido de compilación antes de tiempo y el rendimiento a largo plazo más rápido de JIT adaptativo. Esta técnica puede extenderse a más de dos instancias, por ejemplo, como se describe a continuación con referencia a las Figuras 4 y 5.

Con referencia a la Figura 2, en algunas implementaciones, los usuarios/clientes 201 pueden solicitar ejecutar una aplicación, por ejemplo, ejecutando una pieza de código en un sistema dado, por ejemplo, 205. En algunas implementaciones, puede haber una configuración que designa/etiqueta, por ejemplo, 202, una instancia de la aplicación como primaria 203, y el resto como secundarias 204. En algunas realizaciones, para mantener el mismo registro exacto entre las diferentes instancias de ejecuciones de código (por ejemplo, primaria y secundaria), la aplicación de respaldo usa la misma ruta de código como la instancia primaria. Pero, en algunas realizaciones, algunas etapas en las ejecuciones pueden ser ligeramente diferentes. Por ejemplo, una instancia primaria y una instancia secundaria pueden seguir la misma ruta de código exacta excepto para la instancia primaria que genera una salida después de la ejecución, por ejemplo, 207, mientras que la secundaría no lo hace, por ejemplo, 208. En algunas realizaciones, el procesamiento de generación de salida por la aplicación de respaldo puede ejecutarse a través de toda la ruta de ejecución de la instancia primaria hasta, pero sin incluir, el punto de publicación de la salida, resultando en primera y segunda instancias que se mantienen tan idénticamente cercanas entre sí como sea posible. En algunas realizaciones, las instancias primaria y secundaria pueden no finalizar en un estado exactamente idéntico, sin embargo. Por ejemplo, en sistemas que están codificados en java, si la primera y segunda instancias siguen las mismas rutas de ejecución de código exactas hasta, pero sin incluir, la publicación de las salidas de mensaje de los códigos, pueden tener el flujo de mensaje completo en común y pueden estar en sinc entre sí, pero no estarán exactamente en el mismo estado. Por ejemplo, las peticiones de ejecución 206a a la instancia primaria pueden especificar que necesita producirse una salida final. Por ejemplo, las peticiones de ejecución 206a y 206b pueden ser mensajes POST de Protocolo de Transferencia de Hipertexto (Seguro) ("HTTP(S)") que incluyen una instrucción de "escribir salida" en forma de datos formateados de acuerdo con XML. A continuación se proporciona un listado de ejemplo de una aplicación petición de ejecución 206a y 206b, sustancialmente en forma de un mensaje POST de HTTP(S) que incluye datos formateados con XML:

```
POST /execute_request.php HTTP/1.1
Host: www.primary.com
Content-Type: Application/XML
Content-Length: 867
<?XML version = "1.0" encoding = "UTF-8"?>
<execute request>
   <request id> fdgsbv633dg </instruct_id>
   <timestamp> 2015-05-13 15:43:44</timestamp>
   <application type> primary </application type>
   <prmry_id> PRMRY_1 </prmry_id>
// <application type> backup </application type>
// <backup_size> n=7 <backup_size> // <backup_id> SCNDRY_1 </backup_id>
// <backup_id> SCNDRY_2 </backup_id>
```

```
      ...
      <application inputs>
         <init_val_1> ... </init_val_1>
         <init_val_2> ... </init_val_2>
         ...
      </application inputs>
      <application_name> trading_analytics.exe </application_name>
      <Output_msg> TRUE </output_msg>
      <application_output>
         <format> ... </format>
         ...
      </application output>
   </execute request>
```

En algunas implementaciones, para mejorar rendimiento de tiempo de ejecución de códigos, pueden utilizarse capacidades de compilación justo a tiempo (JIT) de un sistema, y convertir los códigos en códigos máquina. En algunas implementaciones, el compilador JIT puede iniciar la optimización de los códigos como parte de la compilación, si es la primaria, por ejemplo, 207, o la secundaria, por ejemplo, 208. Por ejemplo, compilación JIT puede consolidar líneas de código duplicadas en unas sucintas, y/o eliminar rutas de código ajenas para optimizar más eficientemente las rutas que se están usando más. Por ejemplo, si una ruta de código se ha usado más de un número de veces umbral predeterminado y/o dinámicamente determinado, en algunas realizaciones, puede darse a la misma una mayor prioridad y favorecerse sobre otras rutas de código con menos uso. En algunas realizaciones, la instancia secundaria que puede ejecutarse a través de toda la ruta de ejecución de la instancia primaria hasta el punto de publicación de la salida puede tener una ruta de código que no es la misma que la de la instancia primaria, porque el código de la instancia secundaria que conduce hasta el punto de publicación puede no optimizarse de forma similar como el correspondiente código de la instancia primaria.

En algunas realizaciones, la aplicación secundaria y/o en espera puede convertirse en una instancia primaria cuando se determina que ha fallado la aplicación original primaria y/o activa. Por ejemplo, un servidor 204 puede supervisar el rendimiento de la instancia primaria e iniciar conmutación por error de aplicación cuando detecta fallo de sistema y/o aplicación en la parte de la instancia primaria original, por ejemplo, 209. Una vez que se determina que ha fallado la instancia primaria original, en algunas realizaciones, el servidor puede designar/etiquetar una instancia secundaria de la aplicación en una nueva instancia primaria y dirigir los códigos para ejecución a la nueva instancia primaria, por ejemplo, 210. Por ejemplo, el servidor secundario 204 puede generar un mensaje POST de Protocolo de Transferencia de Hipertexto (Seguro) ("HTTP(S)") que incluye una instrucción de conmutación en forma de datos formateados de acuerdo con XML. A continuación se proporciona un listado de ejemplo de una instrucción de conmutación por error 210, sustancialmente en forma de un mensaje POST de HTTP(S) que incluye datos formateados con XML:

```
   POST /switch_instruction.php HTTP/1.1
   Host: www.switch.com
   Content-Type: Application/XML
   Content-Length: 867
   <?XML version = "1.0" encoding = "UTF-8"?>
   <routing instruct>

      <instruct id> 25jh23jlu0s </instruct id>
      <timestamp> 22/02/2015 15:22:44</timestamp>
      <instructor_id> SCNDRY_2 </instructor_id>
      <switch reason> PRMRY NO RSPNS </switch reason>
      …
      <designation instr>
            <switch> primary = SCNDRY_2 </switch>
            <duration> INDFNT </duration>
            //<duration> PRMRY MSG </duration>
            …
      </designation instr>
      …
   </routing instruct>
```

En algunas realizaciones, después de la instrucción, la ejecución de código puede a continuación dirigirse a la aplicación con la ruta de código óptima en la que el código "publicar" o "escribir" de la aplicación está frío, es decir, no optimizado, por ejemplo, 211. Por ejemplo, la petición de ejecución de código 211 puede ser un mensaje POST de Protocolo de Transferencia de Hipertexto (Seguro) ("HTTP(S)") que incluye una instrucción de "escribir salida" en forma de datos formateados de acuerdo con XML como se ha descrito con referencia a la Figura 2. En algunas realizaciones, el rendimiento del código de ejecución puede ser inferior en comparación con el anteriormente

primario, ya que algunas partes de la ruta de código no están optimizadas. Por ejemplo, en sistemas codificados en java, y una vez que la conmutación se hace a la instancia secundaria con una ruta de código que tiene un código "escrito" frío, java puede ejecutarse a través de número de iteraciones umbral predeterminado y/o dinámicamente determinado del código "escribir" de la ruta de código para optimizar, es decir, "calentar" la ruta de código, y en el proceso ralentiza la aplicación para la duración del periodo de calentamiento, resultando en degradación en rendimiento en comparación con la instancia primaria antes de la conmutación, por ejemplo, 212.

En algunas implementaciones, el sistema de DSFM puede permitir que las aplicaciones secundarias y/o en reposo se ejecuten hasta el final de toda la ruta de código que incluye la publicación de las salidas de mensaje de los códigos. En algunas implementaciones, esto permite que las instancias primaria y secundaria mantengan estados idénticos, y en el evento de fallo con la instancia primaria, la instancia secundaria puede intervenir y proceder con el código de ejecución a lo largo de la misma ruta de código optimizada que habría tomado la anteriormente instancia primaria y, por lo tanto, sin degradación en rendimiento, por ejemplo, 213. Por ejemplo, la instancia secundaria puede permitir que la ruta de procesamiento de mensaje se ejecute hasta el final del segmento de java y que java escriba la salida de mensaje hasta la siguiente etapa, pero a continuación detenga la ruta antes de que comience la siguiente etapa. En estas realizaciones, toda la ruta java puede ejecutarse tanto por las aplicaciones primaria como de respaldo, y ambas aplicaciones pueden mantener un estado idéntico. En algunas realizaciones, si la primaria necesita conmutar por error a la instancia secundaria, la instancia secundaria puede retomar inmediatamente donde la instancia primaria abandonó sin degradación en rendimiento ya que la instancia secundaria tanto se optimiza por java como se "calienta", como la instancia primaria.

La Figura 3 muestra un diagrama de flujo lógico que ilustra ejemplos de transformación de una petición para ejecución de aplicación a través de una realización de un componente de DSFM en una ejecución optimizada y eficiente de la aplicación. En algunas realizaciones, un usuario puede desear ejecutar una aplicación, por ejemplo, 301, e iniciar la ejecución ejecutando una pieza de código de la aplicación. En algunas implementaciones, esta puede ser la primera instancia de la ejecución del código, y la compilación del código puede garantizarse, por ejemplo, 302. Por ejemplo, un código fuente java traducido a un código de bytes puede compilarse por un compilador JIT en un código máquina, por ejemplo, 303. En algunas realizaciones, la compilación puede hacerse dinámicamente y/o puede hacerse a solo una pieza de todo el código (por ejemplo, al método del código que se llamó, etc.). Una vez que la aplicación comienza a ejecutarse, por ejemplo, 304, en algunas realizaciones, java puede comenzar a supervisar la ejecución del código para optimizar la ejecución, por ejemplo, 305. Por ejemplo, java puede esperar un pequeño número de iteraciones antes de identificar métodos, funciones, etc., que se llaman frecuentemente y/o usan como los "puntos calientes", por ejemplo, 306. En algunas implementaciones, una vez que la aplicación se ha ejecutado un suficiente número de veces para determinar estos puntos calientes, los puntos calientes pueden señalizarse para optimización adicional, por ejemplo, 309. Por ejemplo, estos métodos, funciones, etc. pueden hacerse más eficientes consolidando comandos, descartando elementos de código, etc., para mejorar eficiencia y velocidad en la ejecución de la aplicación. En algunas implementaciones, java también puede esperar un número de iteraciones antes de identificar rutas de código que se usan frecuentemente, por ejemplo, 307, y optimizar/"preparar" la aplicación para eficiencia mejorada eliminando rutas de código ajenas que se usan infrecuentemente, por ejemplo, 310. Por ejemplo, después de que una ruta se ha ejecutado 10.000 veces, puede recibir una prioridad más alta, mientras otras rutas que se usan infrecuentemente pueden descartarse en favor de la ruta de mayor prioridad. En algunas implementaciones, una vez que se optimizan puntos calientes, y se identifican rutas de código de prioridad alta, la ejecución de la aplicación puede volverse más eficiente, por ejemplo, 311.

La Figura 4 muestra diagramas de bloque que ilustran un sistema de DSFM ilustrativo que proporciona optimización de aplicación, sincronización y conmutación por error mediante una redundancia de procesos de aplicación. En algunas implementaciones, después de una petición para ejecutar una aplicación, el sistema de DSFM puede iniciar la ejecución de al menos dos instancias de la aplicación, por ejemplo, 401a-n (equivalentemente dos o más aplicaciones idénticas). En algunas realizaciones, puede no haber ninguna designación/etiquetado de estado de estas instancias de las aplicaciones como primaria y secundaria/respaldos, y cada instancia de aplicación puede ejecutarse independientemente para producir salidas de mensajes idénticas o casi idénticas que se escriben en un bus/alambre de mensajería 406. En algunas realizaciones, variaciones menores en procesamiento de sistemas (por ejemplo, software, hardware, etc.) puede provocar estas salidas de mensaje 402 producidas por dos o más instancias de la misma aplicación (es decir, aplicaciones idénticas) a escribirse en el alambre no simultáneamente. En algunas implementaciones, un secuenciador puede procesar los mensajes de llegada y clasificar una salida como la salida de mensaje de la aplicación y el resto como duplicadas. Por ejemplo, un secuenciador 403 puede designar/etiquetar una salida de mensaje escrita en el alambre por una de dos o más instancias de una aplicación como la salida de mensaje de la aplicación a mantener, y etiquetar (y en consecuencia, por ejemplo, descartar) el resto de las salidas de mensaje que proceden de las instancias más lentas de las aplicaciones como duplicadas. En algunas realizaciones, el secuenciador puede utilizar el tiempo de llegada de los mensajes como el criterio para elegir qué mensajes mantener y qué mensajes etiquetar como duplicados. Por ejemplo, el secuenciador puede elegir el primer mensaje a escribir en el alambre como la salida de mensaje de la aplicación y descartar el resto, por ejemplo, 404. En algunas implementaciones, el secuenciador puede utilizar otros criterios en lugar de o en combinación con el tiempo de llegada para determinar el mensaje que debería mantenerse como la salida de mensaje de la aplicación. Por ejemplo, en algunas realizaciones, los criterios pueden incluir el tiempo de llegada del mensaje, el tamaño del mensaje, etc. Por ejemplo, el secuenciador puede emplear el criterio de que el mensaje que

se mantiene como la salida de la aplicación es el primer mensaje que llega cuyo tamaño es menor que algún número umbral de bytes en tamaño (y/o el más pequeño en tamaño, etc.).

En algunas realizaciones, con dos o más aplicaciones idénticas o casi idénticas ejecutándose, los mensajes que se retienen por el secuenciador pueden proceder de diferentes aplicaciones, ya que la aplicación que "gana" en la escritura de un mensaje varía mensaje por mensaje. Por ejemplo, con referencia a la Figura 5, la salida de mensaje final de ejecutar aplicaciones puede reconstruirse por el secuenciador a partir de las salidas de mensaje de las múltiples aplicaciones idénticas o casi idénticas. Por ejemplo, con N aplicaciones ejecutándose, el mensaje de aplicación D puede ser el primero en llegar de un primer lote de N mensajes, por ejemplo, 501. En un segundo lote de N mensajes, los mensajes de otra aplicación (digamos de la aplicación K) pueden ser los primeros en llegar al secuenciador, por ejemplo, 502. En tales realizaciones, el secuenciador 504 puede reconstruir el mensaje a partir de la aplicación D en el primer lote, seguido por el mensaje de la aplicación K en el segundo lote, etc., para llegar a la salida de mensaje final de la aplicación, por ejemplo, 505.

En tales implementaciones en las que se ejecutan múltiples aplicaciones idénticas o casi idénticas, la redundancia puede provocar tráfico de mensajes en el bus/alambre de mensajería. Por ejemplo, con N aplicaciones idénticas o casi idénticas ejecutándose, existirían N idénticas o casi idénticas salidas de mensaje, resultando en tráfico de mensajes creciente para el alambre. A la inversa, la redundancia puede utilizarse para optimización de rendimiento ya que el sistema recoge el beneficio de obtener salidas de mensaje de las aplicaciones más rápidas como se ha descrito anteriormente. En algunas implementaciones, la redundancia puede proporcionar un mecanismo a prueba de fallos en casos en los que una o más aplicaciones fallan, por ejemplo, 405. En tales implementaciones, puede no haber una necesidad de intervenir manualmente y/o automáticamente, ya que existe una o más otras aplicaciones ejecutándose, y pueden obtenerse salidas de mensaje a partir de estas una o más aplicaciones, por ejemplo, 406.

Por lo tanto, diversas realizaciones descritas en este documento pueden aumentar rendimiento de código en lenguajes interpretados y/u otros lenguajes, por ejemplo, con ejecuciones de tiempo compiladas justo en tiempo (JIT) opcionales. Compiladores JIT adaptativos pueden aumentar el rendimiento a largo plazo en comparación con compiladores antes de tiempo y/o JIT simple, a costa de un periodo de rendimiento inicial más lento mientras el código se interpreta y perfila para mejorar la fase de compilación posterior. Ejecutando el mismo código en paralelo en dos o más tiempos de ejecución como dos o más instancias no idénticas, pueden conseguirse los beneficios de compilación JIT optimizada sin sacrificar rendimiento inicial durante la fase interpretada.

Controlador de DSFM

La Figura 6 muestra un diagrama de bloques que ilustra ejemplos de un controlador de DSFM 601. En esta realización, el controlador de DSFM 601 puede servir para agregar, procesar, almacenar, buscar, servir, identificar, ordenar, generar, emparejar y/o facilitar interacciones con un ordenador a través de diversas tecnologías y/u otros datos relacionados.

Los usuarios, por ejemplo, 633a, que pueden ser gente y/u otros sistemas, pueden participar en sistemas de tecnología de información (por ejemplo, ordenadores) para facilitar el procesamiento de información. A su vez, ordenadores emplean procesadores para procesar información; tales procesadores 603 pueden denominarse como unidades de procesamiento central (CPU). Una forma de procesador se denomina como un microprocesador. CPU usan circuitos comunicativos para pasar señales codificadas que actúan como instrucciones para habilitar diversas operaciones. Estas instrucciones pueden ser operacionales y/o instrucciones de datos que contienen y/o referencian otras instrucciones y datos en diversas áreas accesibles por procesador y operables de memoria 629 (por ejemplo, registros, memoria caché, memoria de acceso aleatorio, etc.). Tales instrucciones comunicativas pueden almacenarse y/o transmitirse en lotes (por ejemplo, lotes de instrucciones) como programas y/o componentes de datos para facilitar operaciones deseadas. Estos códigos de instrucciones almacenados, por ejemplo, programas, pueden participar en los componentes de circuito de CPU y otras placa o placas (base) y/o componentes de sistema para realizar operaciones deseadas. Un tipo de programa es un sistema operativo informático, que puede ejecutarse por CPU en un ordenador; el sistema operativo habilita y facilita que usuarios accedan y operen tecnología de información informática y recursos. Algunos recursos que pueden emplearse en sistemas de tecnología de información incluyen: mecanismos de entrada y salida (E/S) a través de los cuales datos pueden pasar dentro y fuera de un ordenador; almacenamiento de memoria en el que pueden guardarse datos; y procesadores mediante los que puede procesarse información. Estos sistemas de tecnología de información pueden usarse para recopilar datos para posterior recuperación, análisis y manipulación, que pueden facilitarse a través de un programa de base de datos. Estos sistemas de tecnología de información proporcionan interfaces que permiten que usuarios accedan y operen diversos componentes de sistema.

En una realización, el controlador de DSFM 601 puede conectarse a y/o comunicar con entidades tales como, pero sin limitación: uno o más usuarios de dispositivos de entrada de usuario 611; dispositivos periféricos 612; un dispositivo de procesador criptográfico 628 opcional; y/o una red de comunicaciones 613. Por ejemplo, el controlador de DSFM 601 puede conectarse a y/o comunicar con usuarios, por ejemplo, 633a, dispositivo o dispositivos de cliente operativos, por ejemplo, 633b, incluyendo, pero sin limitación, ordenador u ordenadores personales, servidor o servidores y/o diversos dispositivo o dispositivos móviles incluyendo, pero sin limitación, teléfono o teléfonos

celulares, teléfono o teléfonos inteligentes (por ejemplo, iPhone®, Blackberry®, teléfonos basados en Android OS, etc.), ordenador u ordenadores de tableta (por ejemplo, Apple iPad™, HP Slate™, Motorola Xoom™, etc.), lector o lectores de libro electrónico (por ejemplo, Amazon Kindle™, libro electrónico Nook™ de Barnes and Noble, etc.), ordenadores portátiles), mini ordenador u ordenadores, ordenador u ordenadores de mano, consola o consolas de juegos (por ejemplo, XBOX Live™, Nintendo® DS, Sony PlayStation® Portable, etc.), escáner o escáneres portátiles y/o similares.

Las redes se entienden comúnmente para incluir la interconexión e interoperación de clientes, servidores y nodos intermediarios en una topología de gráficos. Se ha de observar que el término "servidor" según se usa a lo largo de esta solicitud se refiere generalmente a un ordenador, otro dispositivo, programa o combinación de los mismos que procesa y responde a las peticiones de usuarios remotos a través de una red de comunicaciones. Los servidores sirven su información a "clientes" solicitantes. El término "cliente" como se usa en este documento se refiere generalmente a un ordenador, programa, otro dispositivo, usuario y/o combinación de los mismos que es capaz de procesar y hacer peticiones y obtener y procesar cualquier respuesta de servidores a través de una red de comunicaciones. Un ordenador, otro dispositivo, programa o combinación de los mismos que facilita, procesa información y peticiones y/o reenvía el paso de información desde un usuario de origen a un usuario de destino se denomina comúnmente como un "nodo". Las redes se entienden comúnmente para facilitar la transferencia de información desde puntos de origen a destinos. Un nodo específicamente encargado con el reenvío del paso de información desde una fuente a un destino se llama comúnmente "encaminador". Existen muchas formas de redes tales como Redes de Área Local (LAN), pico redes, Redes de Área Extensa (WAN), redes inalámbricas (WLAN), etc. Por ejemplo, la Internet se acepta generalmente como que es una interconexión de una multitud de redes con lo que clientes y servidores remotos pueden acceder e interoperar entre sí.

El controlador de DSFM 601 puede basarse en sistemas informáticos que pueden incluir, pero sin limitación, componentes tal como: una sistematización informática 602 conectada a la memoria 629.

Un sistema informático 602 habitualmente incluye un reloj 630, unidad de procesamiento central ("CPU" y/o "procesador o procesadores" (estos términos se usan indistintamente a lo largo de toda la divulgación a no ser que se indique lo contrario)) 603, una memoria 629 (por ejemplo, una memoria de solo lectura (ROM) 606, una memoria de acceso aleatorio (RAM) 605, etc.) y/o un bus de interfaz 607, y más frecuentemente, aunque no necesariamente, se interconectan todos y/o comunican a través de un bus de sistema 604 en una o más placa o placas (base) 602 que tienen trayectorias de circuito conductor y/o transportador de otra manera a través de cuyas instrucciones (por ejemplo, señales codificadas binarias) pueden viajar para efectuar comunicaciones, operaciones, almacenamiento, etc. La sistematización informática puede conectarse a una fuente de alimentación 686; por ejemplo, opcionalmente la fuente de alimentación puede ser interna. Opcionalmente, un procesador criptográfico 626 y/o transceptores (por ejemplo, CI) 674 pueden conectarse al bus de sistema. En otra realización, el procesador criptográfico y/o transceptores pueden conectarse como dispositivos periféricos 612 o bien internos y/o externos a través de la E/S de bus de interfaz. A su vez, los transceptores pueden conectarse a la antena o antenas 675, efectuando de este modo transmisión inalámbrica y recepción de diversos protocolos de comunicación y/o sensor; por ejemplo la antena o antenas pueden conectarse a: un chip de transceptor de Texas Instruments WiLink WL1283 (por ejemplo, proporcionando 802.1 In, Bluetooth 3.0, FM, Sistema de Posicionamiento Global (GPS) (permitiendo de este modo que un controlador de DSFM determine su ubicación)); chip de transceptor de Broadcom BCM4329FKUBG (por ejemplo, proporcionando 802.1 In, Bluetooth 2.1 + EDR, FM, etc.), BCM28150 (HSPA+) y BCM2076 (Bluetooth 4.0, GPS, etc.); un chip receptor BCM4750IUB8 de Broadcom (por ejemplo, GPS); un X-Gold 618-PMB9800 de Infineon Technologies (por ejemplo, proporcionando comunicaciones 2G/3G HSDPA/HSUPA); XMM 7160 de Intel (LTE&DC-HSPA), CDMA(2000) de Qualcom, Módem de Datos/Estación Móvil, Snapdragon; y/o similares. El reloj de sistema puede tener un oscilador de cristal y genera una señal base a través de las rutas de circuito de sistematización informática. El reloj puede acoplarse al bus de sistema y a diversos multiplicadores de reloj que aumentarán o disminuirán la frecuencia operativa base para otros componentes interconectados en la sistematización informática. El reloj y diversos componentes en una sistematización informática accionan señales que incorporan información a través de todo el sistema. Tal transmisión y recepción de instrucciones que incorporan información a través de una sistematización informática puede denominarse como comunicaciones. Estas instrucciones comunicativas pueden transmitirse adicionalmente, recibirse y ser la causa de comunicaciones de devolución y/o contestación más allá de la sistematización informática de instantánea para: redes de comunicaciones, dispositivos de entrada, otras sistematizaciones informáticas, dispositivos periféricos y/o similar. Debería entenderse que en realizaciones alternativas, cualesquiera de los componentes anteriores pueden conectarse directamente entre sí, conectarse a la CPU, y/u organizarse en numerosas variaciones empleadas como ilustrativas por diversos sistemas informáticos.

La CPU puede incluir al menos un procesador de datos de alta velocidad adecuado para ejecutar componentes de programa para ejecutar peticiones generadas por usuario y/o sistema. A menudo, los procesadores en sí mismos incorporarán diversas unidades de procesamiento especializadas, tales como, pero sin limitación: unidades de coma flotante, unidades de procesamiento de enteros, controladores (bus) de sistema integrados, unidades operativas lógicas, unidades de control de gestión de memoria, etc. e incluso subunidades de procesamiento especializadas como unidades de procesamiento gráfico, unidades de procesamiento de señal digital y/o similares. Adicionalmente, procesadores pueden incluir memoria direccionable de acceso rápido interna, y se capaces de correlacionar y direccionar la memoria 629 más allá del propio procesador; memoria interna puede incluir, pero sin limitación:

registradores rápidos, diversos niveles de memoria caché (por ejemplo, nivel 1, 2, 3, etc.), RAM, etc. El procesador puede acceder a esta memoria a través del uso de un espacio de dirección de memoria que es accesible a través de dirección de instrucciones, que el procesador puede construir y decodificar permitiendo al mismo acceder a una trayectoria de circuito a un espacio de dirección de memoria específico que tiene un estado/valor de memoria. La CPU puede ser un microprocesador tal como: Athlon de AMD, Duron y/o Opteron; procesadores de ARM clásico (por ejemplo, ARM7/9/11), embebido (Coretx-M/R), de aplicación (Cortex-A) y seguros; DragonBall y PowerPC de IBM y/o Motorola; procesador Cell de IBM y Sony; Atom, Celeron (Móvil), Core (2/Duo/i3/i5/i7), Itanium, Pentium, Xeon y/o Xscale de Intel; y/o procesador o procesadores similares. La CPU interactúa con memoria a través de una instrucción que pasa través de conductos conductores y/o transportadores (por ejemplo, circuitos electrónicos (impresos) y/o ópticos) para ejecutar instrucciones almacenadas (es decir, código de programa). Tal paso de instrucción facilita la comunicación dentro del controlador de DSFM y más allá a través de diversas interfaces. Si los requisitos de procesamiento indican una mayor cantidad de velocidad y/o capacidad, procesadores distribuidos (por ejemplo, sistema/controlador de DSFM distribuido), pueden emplearse de forma similar arquitecturas de ordenadores centrales, multinúcleo, paralelas y/o superordenadores. Como alternativa, si los requisitos de despliegue indican mayor portabilidad, pueden emplearse dispositivos móviles más pequeños (por ejemplo, teléfonos inteligentes, Asistentes Digitales Personales (PDA), etc.).

Dependiendo de la implementación particular, pueden conseguirse características del sistema de DSFM implementando un microcontrolador tal como el microcontrolador R8051XC2 de CAST; MCS 51 de Intel (es decir, microcontrolador 8051); y/o similares. También, para implementar ciertas características del sistema de DSFM, algunas implementaciones de características pueden basarse en componentes embebidos, tal como: Circuito Integrado Específico de Aplicación ("ASIC"), Procesamiento de Señal Digital ("DSP"), Campo de Matriz de Puertas Programables ("FPGA") y/o tecnología embebida similar. Por ejemplo, cualquiera de la colección de componentes de DSFM (distribuidos o de otra manera, por ejemplo IMAS 341, etc.) y/o características pueden implementarse a través del microprocesador y/o a través de componentes embebidos; por ejemplo, a través de ASIC, coprocesador, DSP, FPGA y/o similar. Como alternativa, algunas implementaciones del sistema de DSFM pueden implementarse con componentes embebidos que se configuran y usan para conseguir una diversidad de características o procesamiento de señales.

Dependiendo de la implementación particular, los componentes embebidos pueden incluir soluciones de software, soluciones de hardware y/o alguna combinación de ambas soluciones de hardware/software. Por ejemplo, las características de sistema de DSFM analizadas en este documento pueden conseguirse a través de la implementación de FPGA, que son dispositivos de semiconductores que contienen componentes lógicos programables llamados "bloques lógicos", e interconexiones programables, tal como la serie Virtex de FPGA de alto rendimiento y/o la serie Spartan de bajo coste fabricados por Xilinx. Bloques lógicos e interconexiones pueden programarse por el cliente o diseñador, después de que el FPGA se fabrique, para implementar cualquiera de las características de sistema de DSFM. Una jerarquía de interconexiones programables permite que bloques lógicos se interconecten según se necesiten por el diseñador/administrador de sistema de DSFM, en cierto modo como una placa de pruebas programable de un solo chip. Los bloques lógicos de un FPGA pueden programarse para realizar la operación de puertas lógicas básicas tal como AND y XOR, u operadores combinacionales más complejos tal como decodificadores u operaciones matemáticas simples. En la mayoría de FPGA, los bloques lógicos también incluyen elementos de memoria, que pueden ser circuitos biestables o bloques más completos de memoria. En algunas circunstancias, el sistema de DSFM puede desarrollarse en FPGA regulares y a continuación migrarse a una versión fija que se asemeja más a implementaciones de ASIC. Implementaciones alternativas o de coordinación pueden migrar una o más características de controlador de DSFM a un ASIC final en lugar de o además de FPGA. Dependiendo de la implementación todos componentes embebidos y microprocesadores anteriormente mencionados pueden considerarse la "CPU" y/o "procesador" para el sistema de DSFM.

La fuente de alimentación 686 puede ser de cualquier forma estándar para proporcionar potencia a dispositivos de placa de circuito electrónico pequeños tales como las siguientes pilas: alcalina, hidruro de litio, ion litio, polímero de litio, níquel cadmio, células solares y/o similares. También pueden usarse otros tipos de fuentes de alimentación de CA o CC. En el caso de células solares, en una realización, el caso proporciona una abertura a través de la cual la célula solar puede capturar energía fotónica. La pila 686 se conecta a al menos uno de los componentes posteriores interconectados del sistema de DSFM proporcionando de este modo una corriente eléctrica a todos sus componentes interconectados. En un ejemplo, la fuente de alimentación 686 se conecta al componente de bus de sistema 604. En una realización alternativa, se proporciona una fuente de alimentación externa 686 a través de una conexión a través de la interfaz de E/S 608. Por ejemplo, una conexión de USB y/o IEEE 1394 transporta tanto datos y potencia a través de la conexión y, por lo tanto, es una fuente de alimentación adecuada.

El bus o buses de interfaz 607 pueden aceptar, conectar y/o comunicarse a un número de adaptadores de interfaz, frecuentemente, aunque no necesariamente en forma de tarjetas de adaptador, tal como pero sin limitación: interfaces de Entrada Salida (E/S) 608, interfaces de almacenamiento 609, interfaces de red 610 y/o similares. Opcionalmente, interfaces de procesador criptográfico 627 pueden conectarse de forma similar al bus de interfaz. El bus de interfaz proporciona las comunicaciones de adaptadores de interfaz entre sí, así como con otros componentes de la sistematización informática. Los adaptadores de interfaz se adaptan para un bus de interfaz compatible. Los adaptadores de interfaz pueden conectarse al bus de interfaz a través de una arquitectura de

expansión y/o ranura. Pueden emplearse diversas arquitecturas de expansión y/o ranura, tales como, pero sin limitación: Puerto de Gráficos Acelerado (AGP), Bus de Tarjeta, ExpressCard, Arquitectura Estándar de la Industria (Extendida) ((E)ISA), Arquitectura Micro Canal (MCA), NuBus, Interconexión de Componentes Periféricos (Extendida) (PCI(X)), PCI Express, Asociación Internacional de Tarjetas de Memoria para Ordenadores Personales (PCMCIA), Thunderbolt y/o similar.

Las interfaces de almacenamiento 609 pueden aceptar, comunicarse y/o conectarse a un número de dispositivos de almacenamiento tales como, pero sin limitación: dispositivos de almacenamiento 614, dispositivos de disco extraíbles y/o similares. Interfaces de almacenamiento pueden emplear protocolos de conexión tales como, pero sin limitación: (Ultra) Tecnología Avanzada de Contacto (en Serie) (Interfaz de Paquetes) ((Ultra) ATA (en Serie) (PI)), Electrónica de Unidades Integrada (Mejorada) ((E)IDE), Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) 1394, Ethernet, canal de fibra, Interfaz para Pequeños Sistemas Informáticos (SCSI), Thunderbolt, Bus Serial Universal (USB) y/o similares.

Las interfaces de red 610 pueden aceptar, comunicarse y/o conectarse a una red de comunicaciones 613. A través de una red de comunicaciones 613, el controlador de DSFM es accesible a través de clientes remotos 633b (por ejemplo, ordenadores con navegadores web) por los usuarios 633a. Las interfaces de red pueden emplear protocolos de conexión tal como, pero sin limitación: conexión directa, Ethernet (gruesa, delgada, par trenzado 10/100/1000 Base T y/o similar), Anillo de Testigos, conexión inalámbrica tal como IEEE 802.11a-x y/o similar. Si los requisitos de procesamiento indicasen una mayor cantidad de velocidad y/o capacidad, los controladores de red distribuidos (por ejemplo, sistema de DSFM distribuido), pueden emplearse de forma similar arquitecturas para agrupar, realizar equilibrio de carga y/o de otra manera aumentar el ancho de banda comunicativo requerido por el controlador de DSFM. Una red de comunicaciones puede ser una cualquiera y/o la combinación de las siguientes: una interconexión directa; la Internet; una Red de Área Local (LAN); una Red de Área Metropolitana (MAN); Misiones que Operan como Nodos en la Internet (OMNI); una conexión personalizada segura; una Red de Área Extensa (WAN); una red inalámbrica (por ejemplo, empleando protocolos tal como, pero sin limitación a un Protocolo de Aplicación Inalámbrica (WAP), modo I y/o similar); y/o similares. Una interfaz de red puede considerarse como una forma especializada de una interfaz de entrada salida (E/S). Además, pueden usarse múltiples interfaces de red 610 para acoplarse con diversos tipos de red de comunicaciones 613. Por ejemplo, pueden emplearse múltiples interfaces de red para permitir la comunicación a través de redes de difusión, multidifusión o unidifusión.

Las interfaces de Entrada Salida (E/S) 608 pueden aceptar, comunicarse y/o conectarse a dispositivos de entrada de usuario 611, dispositivos periféricos 612, dispositivos de procesador criptográfico 628 y/o similares. Las E/S pueden emplear protocolos de conexión tales como, pero sin limitación: audio: analógico, digital, monofónico, RCA, estéreo y/o similares; datos: Bus de Escritorio de Apple (ADB), Bluetooth, IEEE 1394a-b, en serie, Bus Serial Universal (USB); infrarrojo; palanca de mandos; teclado; midi; óptico; PC AT; PS/2; paralelo; radio; interfaz de vídeo: Conector de Escritorio de Apple (ADC), BNC, coaxial, componente, compuesto, digital, Display-port, Interfaz Visual Digital (DVI), Interfaz Multimedia de Alta Definición (HDMI), RCA, antenas de RF, S-Video, VGA y/o similares; transceptores inalámbricos: 802.11a/b/g/n/x; Bluetooth; celular (por ejemplo, acceso múltiple por división de código (CDMA), acceso de alta velocidad por paquetes (HSPA(+)), acceso de paquetes de enlace descendente a alta velocidad (HSDPA), sistema global para comunicaciones móviles (GSM), evolución a largo plazo (LTE), WiMax, etc.); y/o similares. Un dispositivo de salida puede ser un visualizador de vídeo, que puede tomar la forma de un monitor basado en Tubo de Rayos Catódicos (CRT), Pantalla de Cristal Líquido (LCD), Diodo Emisor de Luz (LED), Diodo Orgánico Emisor de Luz (OLED), Plasma y/o similar, con una interfaz (por ejemplo, VGA, circuitería de DVI y cable) que acepta señales desde una interfaz de vídeo. La interfaz de vídeo compone información generada por una sistematización informática y genera señales de vídeo basándose en la información compuesta en una trama de memoria de vídeo. Otro dispositivo de salida es un televisor, que acepta señales desde una interfaz de vídeo. A menudo, la interfaz de vídeo proporciona la información de vídeo compuesta a través de una interfaz de conexión de vídeo que acepta una interfaz de visualización vídeo (por ejemplo, un conector de vídeo compuesto RCA que acepta un cable de vídeo compuesto RCA; un conector de DVI que acepta un cable de visualización de DVI, HDMI, etc.).

Los dispositivos de entrada de usuario 611 a menudo son un tipo del dispositivo periférico 612 (véase a continuación) y pueden incluir: lectores de tarjetas, llaves, lectores de huellas digitales, guantes, tabletas gráficas, palancas de mandos, teclados, micrófonos, ratón, controles remotos, lectores de retina, pantallas táctiles (por ejemplo, capacitivas, resistivas, etc.), bolas de mando, paneles táctiles, sensores (por ejemplo, acelerómetros, iluminación ambiental, GPS, giroscopios, proximidad, etc.), lápices ópticos y/o similares.

Los dispositivos periféricos 612 pueden conectarse y/o comunicarse con E/S y/u otras instalaciones similares tales como interfaces de red, interfaces de almacenamiento, directamente al bus de interfaz, bus de sistema, la CPU y/o similares. Los dispositivos periféricos pueden ser externos, internos y/o parte del controlador de DSFM. Los dispositivos periféricos puede incluir: antenas, dispositivos de audio (por ejemplo, en línea, fuera de línea, entrada de micrófono, altavoces, etc.), cámaras (por ejemplo, fija, vídeo, cámara web, etc.), llaves (por ejemplo, para protección de copia, asegurar transacciones seguras con una firma digital y/o similar), procesadores externos (para capacidades añadidas; por ejemplo, los dispositivos criptográficos 628), dispositivos de realimentación de fuerza (por ejemplo, motores de vibración), dispositivos de comunicación de campo cercano (NFC), interfaces de red, impresoras, identificadores de frecuencia de radio (RFID), escáneres, dispositivos de almacenamiento, transceptores

(por ejemplo, celular, GPS, etc.), dispositivos de vídeo (por ejemplo, gafas, monitores, etc.), fuentes de vídeo, visores y/o similares. Los dispositivos periféricos a menudo incluyen tipos de dispositivos de entrada (por ejemplo, micrófonos, cámaras, etc.).

Se ha de observar que aunque pueden emplearse dispositivos de entrada de usuario y dispositivos periféricos, el controlador de DSFM puede incorporarse como un dispositivo embebido, especializado y/o sin monitor (es decir, sin cabeza), en el que el acceso se proporcionaría a través de una conexión de interfaz de red.

Unidades criptográficas tal como, pero sin limitación, microcontroladores, procesadores 626, interfaces 627 y/o dispositivos 628 pueden fijarse y/o comunicarse con el controlador de DSFM. Puede usarse un microcontrolador MC68HC16, fabricado por Motorola Inc., para y/o dentro de unidades criptográficas. El microcontrolador MC68HC16 utiliza una instrucción de multiplicación y acumulación de 16 bits en la configuración de 16 MHz y requiere menos de un segundo para realizar una operación de clave privada RSA de 512 bits. Las unidades criptográficas soportan la autenticación de comunicaciones de agentes que interactúan, así como permiten transacciones anónimas. Las unidades criptográficas también pueden configurarse como parte de la CPU. También pueden usarse microcontroladores y/o procesadores equivalentes. Otros procesadores criptográficos especializados disponibles comercialmente incluyen: CryptoNetX y otros Procesadores de Seguridad de Broadcom; nShield de nCipher (por ejemplo, Solo, Connect, etc.), serie Luna PCI de SafeNet (por ejemplo, 7100); Roadrunner 184 de 40 Mhz de Semaphore Comunicaciones; sMIP (por ejemplo, 208956); Aceleradores Criptográficos de Sun (por ejemplo, Placa PCIe de Acelerador 6000, Tarjeta hija de Acelerador 500); / (por ejemplo, L2100, L2200, U2400) línea, que es capaz de realizar 500+ MB/s de instrucciones criptográficas; 6868 de 33 Mhz de VLSI Technology; y/o similares.

En general, cualquier mecanización y/o realización que permite que un procesador afecte al almacenamiento y/o recuperación de información se considera como memoria 629. Sin embargo, la memoria es una tecnología y recurso fungibles, por lo tanto, puede emplearse cualquier número de realizaciones de memoria en lugar de o en colaboración entre sí. Debe apreciarse que el controlador de DSFM y/o una sistematización informática puede emplear diversas formas de memoria 629. Por ejemplo, puede configurarse una sistematización informática en la que la operación de memoria de CPU en chip (por ejemplo, registradores), RAM, ROM y cualquier otro dispositivo de almacenamiento se proporciona mediante un mecanismo de cinta de papel perforada y tarjeta de papel perforada; sin embargo, una realización de este tipo resultaría en una tasa de operación extremadamente lenta. En una configuración, la memoria 629 incluirá ROM 606, RAM 605 y un dispositivo de almacenamiento 614. Un dispositivo de almacenamiento 614 puede emplear cualquier número de dispositivos/sistemas de almacenamiento informáticos. Los dispositivos de almacenamiento pueden incluir un cilindro; una unidad de disco magnético (fija y/o extraíble); una unidad magnetoóptica; una unidad óptica (es decir, Bluray, CD ROM/RAM/Grabable (R)/Regrabable (RW), DVD R/RW, HD DVD R/RW etc.); una matriz de dispositivos (por ejemplo, Matriz Redundante de Discos Independientes (RAID)); dispositivos memoria de estado sólido (memoria USB, unidades de estado sólido (SSD), etc.); otros medios de almacenamiento legibles por procesador; y/u otros dispositivos similares. Por lo tanto, una sistematización informática generalmente requiere y hace uso de memoria.

La memoria 629 puede contener una colección de componentes de programa y/o de base de datos y/o datos tales como, pero sin limitación: componente o componentes de sistema operativo 615 (sistema operativo); componente o componentes de servidor de información 616 (servidor de información); componente o componentes de interfaz de usuario 617 (interfaz de usuario); componente o componentes de navegador web 618 (navegador web); base o bases de datos 619; componente o componentes de servidor de correo 621; componente o componentes de cliente de correo 622; componente o componentes de servidor criptográfico 620 (servidor criptográfico); el componente o componentes de DSFM 635; y/o similares (es decir, colectivamente una colección de componentes). Estos componentes pueden almacenarse y accederse desde los dispositivos de almacenamiento y/o desde dispositivos de almacenamiento accesibles a través de un bus de interfaz. Aunque componentes de programa no convencionales tales como los de la colección de componentes, pueden almacenarse en un dispositivo de almacenamiento local 614, también pueden cargarse y/o almacenarse en memoria tal como: dispositivos periféricos, RAM, instalaciones de almacenamiento remotas a través de una red de comunicaciones, ROM, diversas formas de memoria y/o similares.

El componente de sistema operativo 615 es un componente de programa ejecutable que facilita la operación del controlador de DSFM. El sistema operativo puede facilitar acceso de E/S, interfaces de red, dispositivos periféricos, dispositivos de almacenamiento y/o similares. El sistema operativo puede ser un sistema altamente tolerante a fallos, escalable y seguro tal como: Apple Macintosh OS X (servidor); AT&T Plan 9; Be OS; distribuciones Unix y de tipo Unix (tal como UNIX de AT&T; variaciones de Distribución de Software de Berkley (BSD) tal como FreeBSD, NetBSD, OpenBSD y/o similares; distribuciones Linux tales como Red Hat, Ubuntu y/o similares); y/o sistemas operativos similares. Sin embargo, también pueden emplearse sistemas operativos más limitados y/o menos seguros tales como Apple Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows 2000/2003/3.1/95/98/CE/Millenium/NT/Vista/XP (servidor), Palm OS y/o similares. Además, pueden emplearse sistemas operativos móviles tales como iOS de Apple, Android de Google, WebOS de Hewlett Packard, Microsoft Windows Mobile y/o similares. Cualquiera de estos sistemas operativos puede embeberse dentro del hardware del controlador de DSFM, y/o almacenarse/cargarse en memoria/almacenamiento. Un sistema operativo puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo, y/o similares. Más frecuentemente, el sistema operativo se comunica con otros componentes de programa, interfaces de usuario

y/o similares. Por ejemplo, el sistema operativo puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario y/o comunicaciones de datos, peticiones y/o respuestas. El sistema operativo, una vez ejecutado por la CPU, puede habilitar la interacción con redes de comunicaciones, datos, E/S, dispositivos periféricos, componentes de programa, memoria, dispositivos de entrada de usuario, y/o similares. El sistema operativo puede proporcionar protocolos de comunicaciones que permiten que el controlador de DSFM se comunique con otras entidades a través de una red de comunicaciones 613. El controlador de DSFM puede usar diversos protocolos de comunicación como un mecanismo de transporte de subportadora para interacción, tal como, pero sin limitación: multidifusión, TCP/IP, UDP, unidifusión, y/o similares.

Un componente de servidor de información 616 es un componente de programa almacenado que se ejecuta por una CPU. El servidor de información puede ser un servidor de información de Internet tal como, pero sin limitación, Apache de Apache Software Foundation, Internet Information Server de Microsoft y/o similares. El servidor de información puede permitir la ejecución de componentes de programa a través de instalaciones tales como Página de Servidor Activa (ASP), ActiveX, (ANSI) (Objective-) C (++), C# y/o .NET, guiones de Interfaz de Pasarela Común (CGI), Lenguaje de Marcas de Hipertexto (HTML) dinámico (D), FLASH, Java, JavaScript, Lenguaje Práctico de Extracción e Informes (PERL), Preprocesador de Hipertexto (PHP), conductos, Python, protocolo de aplicación inalámbrica (WAP), WebObjects y/o similares. El servidor de información puede soportar protocolos de comunicaciones seguros tales como, pero sin limitación, Protocolo de Transferencia de Archivos (FTP); Protocolo de Transferencia de Hipertexto (HTTP); Protocolo de Transferencia de Hipertexto Seguro (HTTPS), Capa de Conexión Segura (SSL), protocolos de mensajería (por ejemplo, Instant Messenger (AIM) de America Online (AOL), IMessage de Apple, Intercambio de Aplicación (APEX), ICQ, Chat Interactivo de Internet (IRC), Microsoft Network (MSN) Messenger Service, Protocolo de Presencia y Mensajería Instantánea (PRIM), Protocolo de Iniciación de Sesión (SIP) del Grupo Especial sobre Ingeniería de Internet (IETF), SIP para Extensiones de Aprovechamiento de Presencia y Mensajería Instantánea (SIMPLE), Protocolo de Presencia y Mensajería Extensible basado en XML (XMPP) abierto (es decir, Jabber o Servicio de Presencia y Mensajería Instantánea (IMPS) de la Alianza Móvil Abierta (OMA)), Yahoo! Instant Messenger Service y/o similar. El servidor de información proporciona resultados en forma de páginas web a navegadores web, y permite la generación manipulada de las páginas web a través de interacción con otros componentes de programa. Después de que una porción de resolución de Sistema de Nombre de Dominio (DNS) de una solicitud de HTTP se resuelve a un servidor de información particular, el servidor de información resuelve peticiones para información en ubicaciones especificadas en el controlador de DSFM basándose en el resto de la solicitud de HTTP. Por ejemplo, una petición tal como http://123.124.125.126/myInformation.html podría tener la porción de IP de la petición "123.124.125.126" resuelta por un servidor de DNS a un servidor de información en esa dirección IP; ese servidor de información podría evaluar a su vez adicionalmente la solicitud de HTTP para la porción "/myInformation.html" de la petición y resolver la misma a una ubicación en memoria que contiene la información "my-Information.html". Adicionalmente, pueden emplearse otros protocolos de servicio de información a través de diversos puertos, por ejemplo, comunicaciones FTP a través de puerto 21 y/o similares. Un servidor de información puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. Más frecuentemente, el servidor de información comunica con la base de datos de DSFM 619, sistemas operativos, otros componentes de programa, interfaces de usuario, navegadores web y/o similares.

Puede conseguirse acceso a la base de datos de DSFM a través de un número de mecanismos de puente de base de datos tales como a través de lenguajes de guiones como se enumeran a continuación (por ejemplo, CGI) y a través de canales de comunicación inter aplicación como se enumeran a continuación (por ejemplo, CORBA, WebObjects, etc.). Cualquier petición de datos a través de un explorador web se evalúan a través del mecanismo de puente en gramática apropiada según se requiere por el DSFM. En una realización, el servidor de información proporcionaría un formulario web accesible por un explorador web. Las entradas hechas en los campos suministrados en el formulario web se etiquetan como que se han introducido en los campos particulares, y analizan como tales. Los términos introducidos se analizan a continuación junto con las etiquetas de campo, que actúan para ordenar al analizador que genere consultas dirigidas a tablas y/o campos apropiados. En una realización, el analizador puede generar consultas en SQL estándar instanciando una cadena de búsqueda con los comandos de unir/seleccionar apropiados basándose en las entradas de texto etiquetadas, en la que el comando resultante se proporciona a través del mecanismo de puente al DSFM como una consulta. Tras generar resultados de consulta a partir de la consulta, los resultados se pasan a través del mecanismo de puente, y pueden analizarse para formateo y generación de una nueva página web de resultados por el mecanismo de puente. Una nueva página web de resultados de este tipo se proporciona a continuación al servidor de información, que puede suministrar la misma al navegador web solicitante.

También, un servidor de información puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de datos, peticiones y/o respuestas.

Interfaces informáticas en algunos aspectos son similares a interfaces de operaciones de automóvil. Elementos de interfaz de operaciones de automóvil, tales como, volantes, palancas de cambio y velocímetros facilitan el acceso, operación y visualización de recursos de automóvil y estado. Elementos de interfaz de interacción de ordenador tales como casilla de verificación, cursores, menús, desplazadores y ventanas (colectivamente y comúnmente denominados como miniaplicaciones) facilitan de forma similar el acceso, capacidades, operación y visualización de

datos y recursos de hardware informático y sistema operativo, y estado. Las interfaces de operación se llaman comúnmente interfaces de usuario. Interfaces gráficas de usuario (GUI) tales como Aqua del Sistema Operativo de Apple Macintosh y Cocoa Touch de iOS, OS/2 de IBM, UI de Android Móvil de Google, Windows 2000/2003/3.1/95/98/CE/Millenium/Mobile/NT/XP/Vista/7/8 (es decir, Aero, Metro) de Microsoft, X-Windows de Unix (por ejemplo, que puede incluir librerías y capas de interfaz gráfica de Unix adicionales tales como Entorno de Escritorio K (KDE), mythTV y Entorno de Modelo de Objeto de Red GNU (GNOME)), librerías de interfaz web (por ejemplo, ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, etc. librerías de interfaz tales como, pero sin limitación, Dojo, jQuery(UI), MooTools, Prototype, script.aculo.us, SWFObject, Interfaz de Usuario de Yahoo!, cualquiera de las cuales puede usarse y) proporcionan una línea base y medios de acceso y visualización de información gráficamente a usuarios.

Un componente de interfaz de usuario 617 es un componente de programa almacenado que se ejecuta por una CPU. La interfaz de usuario puede ser una interfaz gráfica de usuario según se proporciona por, con y/o sobre sistemas operativos y/o entornos operativos tales como ya se han analizado. La interfaz de usuario puede permitir la visualización, ejecución, interacción, manipulación y/u operación de componentes de programa y/o instalaciones de sistema a través de instalaciones textuales y/o gráficas. La interfaz de usuario proporciona una instalación a través de la cual usuarios pueden afectar, interactuar y/u operar un sistema informático. Una interfaz de usuario puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. Más frecuentemente, la interfaz de usuario comunica con sistemas operativos, otros componentes de programa y/o similares. La interfaz de usuario puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de datos, peticiones y/o respuestas.

Un componente de navegador web 618 es un componente de programa almacenado que se ejecuta por una CPU. El explorador web puede ser una aplicación de visualización de hipertexto tal como Chrome (Móvil) de Google, Microsoft Internet Explorer, Netscape Navigator, Safari (Móvil) de Apple, objetos web de navegador embebidos tales como a través de clases de objeto Cocoa (Táctil) de Apple y/o similares. La navegación web segura puede suministrarse con encriptación de 128 bits (o mayor) por medio de HTTPS, SSL y/o similares. Navegadores web permitiendo la ejecución de componentes de programa a través de instalaciones tal como ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, API de extensiones de navegador (por ejemplo, Chrome, FireFox, Internet Explorer, Extensión de Safari y/o API similares) y/o similares. Los navegadores web y herramientas de acceso a información similares pueden integrarse en PDA, teléfonos celulares, teléfonos inteligentes y/o otros dispositivos móviles. Un explorador web puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. Más frecuentemente, el explorador web comunica con servidores de información, sistemas operativos, componentes de programa integrados (por ejemplo, extensiones) y/o similares; por ejemplo, puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de datos, peticiones y/o respuestas. También, en lugar de un explorador web y servidor de información, puede desarrollarse una aplicación combinada para realizar operaciones similares de ambos. La aplicación combinada efectuaría de forma similar la obtención y la provisión de información a usuarios, agentes de usuario y/o similares desde los nodos equipados con DSFM. La aplicación combinada puede ser ineficaz en sistemas que emplean navegadores web estándar.

Un componente de servidor de correo 621 es un componente de programa almacenado que se ejecuta por una CPU 603. El servidor de correo puede ser un servidor de correo de Internet tal como, pero sin limitación Servidor de Mail de Apple (3), dovecot, sendmail, Microsoft Exchange y/o similares. El servidor de correo puede permitir la ejecución de componentes de programa a través de instalaciones tales como ASP, ActiveX, (ANSI) (Objective-) C (++), C# y/o .NET, guiones de CGI, Java, JavaScript, PERL, PHP, conductos, Python, WebObjects y/o similares. El servidor de correo puede soportar protocolos de comunicaciones tales como, pero sin limitación: Protocolo de Acceso a Mensaje de Internet (IMAP), Interfaz de Programación de Aplicación de Mensajería (MAPI)/Microsoft Exchange, Protocolo de Oficina Postal (POP3), Protocolo de Transferencia de Correo Simple (SMTP) y/o similares. El servidor de correo puede encaminar, reenviar y procesar mensajes de correo entrantes y salientes que se ha enviado, retransmitido y/o están atravesando de otra manera a través de y/o al sistema de DSFM.

Acceso al sistema de correo de DSFM puede conseguirse a través de un número de API ofrecidas por los componentes de servidor web individuales y/o el sistema operativo.

También, un servidor de correo puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de datos, peticiones, información y/o respuestas.

Un componente de cliente de correo 622 es un componente de programa almacenado que se ejecuta por una CPU 603. El cliente de correo puede ser una aplicación de visualización de correo tal como Mail (Móvil) de Apple, Microsoft Entourage, Microsoft Outlook, Microsoft Outlook Express, Mozilla, Thunderbird y/o similares. Los clientes de correo pueden soportar un número de protocolos de transferencia, tales como: IMAP, Microsoft Exchange, POP3, SMTP y/o similares. Un cliente de correo puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. Más frecuentemente, el cliente de correo comunica con mail servidores, sistemas operativos, otros clientes de correo y/o similares; por ejemplo, puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de

datos, peticiones, información y/o respuestas. En general, el cliente de correo proporciona una instalación para componer y transmitir mensajes de correo electrónico.

Un componente de servidor criptográfico 620 es un componente de programa almacenado que se ejecuta por una CPU 603, procesador criptográfico 626, interfaz de procesador criptográfico 627, dispositivo de procesador criptográfico 628 y/o similares. Interfaces de procesador criptográfico proporcionarán expedición de peticiones de encriptación y/o desencriptación por el componente criptográfico; sin embargo, el componente criptográfico, como alternativa, puede ejecutarse en una CPU. El componente criptográfico permite la encriptación y/o desencriptación de datos proporcionados. El componente criptográfico permite encriptación y/o desencriptación tanto simétricas como asimétricas (por ejemplo, Privacidad Bastante Buena (PGP)). El componente criptográfico puede emplear técnicas criptográficas tales como, pero sin limitación: certificados digitales (por ejemplo, marco de autenticación X.509), firmas digitales, firmas duales, envolvente, protección de acceso con contraseña, gestión de clave pública y/o similares. El componente criptográfico facilitará numerosos protocolos de seguridad (encriptación y/o desencriptación) tales como, pero sin limitación: suma de control, Norma de Encriptación de Datos (DES), Encriptación de Curva Elíptica (ECC), Algoritmo Internacional de Encriptación de Datos (IDEA), Algoritmo 5 de resumen de mensaje (MD5, que es una operación de troceo de una dirección), contraseñas, Cifrado de Rivest (RC5), Rijndael, RSA (que es un sistema de encriptación y autenticación de Internet que usa un algoritmo desarrollado en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman), Algoritmo de Función de Troceo Seguro (SHA), Capa de Conexión Segura (SSL), Protocolo de Transferencia de Hipertexto Seguro (HTTPS) y/o similares. Empleando tales protocolos de seguridad de encriptación, el sistema de DSFM puede encriptar todas las comunicaciones entrantes y/o salientes y pueden servir como nodo dentro de una Red Privada Virtual (VPN) con una red de comunicaciones más ancha. El componente criptográfico facilita el proceso de "autorización de seguridad" con lo que acceso a un recurso se inhibe mediante un protocolo de seguridad en el que el componente criptográfico efectúa acceso autorizado al recurso seguro. Además, el componente criptográfico puede proporcionar identificadores únicos de contenido, por ejemplo, empleando un troceo de MD5 para obtener una firma única para un archivo de audio digital. Un componente criptográfico puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. El componente criptográfico soporta esquemas de encriptación que permiten la transmisión segura de información a través de una red de comunicaciones para habilitar que uno o más componentes DSFM para participar en transacciones seguras si así se desea. El componente criptográfico facilita el acceso seguro de recursos en el sistema de DSFM y facilita el acceso de recursos seguros en sistemas remotos; es decir, puede actuar como un cliente y/o servidor de recursos seguros. Más frecuentemente, el componente criptográfico comunica con servidores de información, sistemas operativos, otros componentes de programa y/o similares. El componente criptográfico puede contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de datos, peticiones y/o respuestas.

El componente de base de datos de DSFM 619 puede incorporarse en una base de datos y sus datos almacenados. La base de datos es un componente de programa almacenado, que se ejecuta por la CPU; configurando la porción de componente de programa almacenado la CPU para procesar los datos almacenados. La base de datos puede ser cualquiera de un número de bases de datos tolerante a fallos, relacional, escalable, segura tales como DB2, MySQL, Oracle, Sybase y/o similares. Las bases de datos relacionales son una extensión de un archivo plano. Las bases de datos relacionales constan de una serie de tablas relacionadas. Las tablas se interconectan a través de un campo clave. El uso del campo clave permite la combinación de las tablas indexando contra el campo clave; es decir, los campos clave actúan como puntos de pivote dimensionales para combinar información de diversas tablas. Relaciones generalmente identifican enlaces mantenidos entre tablas emparejando claves primarias. Claves primarias representan campos que inequívocamente identifican las filas de una tabla en una base de datos relacional. Más precisamente identifican inequívocamente filas de una tabla en el lado "uno" de una relación de "uno a muchos".

Como alternativa, la base de datos de DSFM puede implementarse usando diversas estructuras de datos estándar, tal como una matriz, troceo, lista (enlazada), estructura, archivo de texto estructurado (por ejemplo, XML), tabla y/o similares. Tales estructuras de datos pueden almacenarse en memoria y/o en ficheros (estructurados). En otra alternativa, puede usarse una base de datos orientada a objetos, tal como Frontier, ObjectStore, Poet, Zope y/o similar. Bases de datos de objetos pueden incluir un número de colecciones de objetos que se agrupan y/o enlazan juntos mediante atributos comunes; pueden relacionarse con otras colecciones de objetos mediante algunos atributos comunes. Bases de datos orientadas a objetos funcionan de forma similar a bases de datos relacionales con la excepción de que los objetos no solo son piezas de datos sino que pueden tener otros tipos de capacidades encapsuladas dentro de un objeto dado. Si la base de datos de DSFM se implementa como una estructura de datos, el uso de la base de datos de DSFM 619 puede integrarse en otro componente tal como el componente de DSFM 635. También, la base de datos puede implementarse como una mezcla de estructuras de datos, objetos y estructuras relacionales. Bases de datos pueden consolidarse y/o distribuirse en innumerables variaciones a través de técnicas de procesamiento de datos estándar. Porciones de bases de datos, por ejemplo, tablas, pueden exportarse y/o importarse y, por lo tanto, descentralizarse y/o integrarse.

En una realización, el componente de base de datos 619 incluye varias tablas 619a-f. Una tabla de Usuarios 619a puede incluir campos tales como, pero no limitados a: user_id, ssn, dob, first_name, last_name, age, state,

address_firstline, address_secondline, zipcode, devices_list, contact_info, contact_type, alt_contact_info, alt_contact_type y/o similares. La tabla de Usuarios puede soportar y/o rastrear múltiples cuentas de entidades en un sistema de DSFM. Una tabla de Clientes 619b puede incluir campos tales como, pero no limitados a: device_ID, device_name, device_IP, device_MAC, device_type, device_model, device_version, device_OS, device_apps_list, device_securekey y/o similares. Una tabla de Aplicaciones 619c puede incluir campos tales como, pero no limitados a: application_ID, application_name, application_type, application_backup_list, application_sync y/o similares. Una tabla de Mensajes 619d puede incluir campos tales como, pero no limitados a: msg_id, msg_application, timestamp, msg_details_list, message_size, message_origin, message_synchronization y/o similares. Una tabla de Conmutación por error 619e puede incluir campos tales como, pero no limitados a: failover_ID, failover_timestamp, primary_application, secondary_application, failover_check_time y/o similares. Una tabla de Rutas de código 619f puede incluir campos tales como, pero no limitados a: codepath_ID, codepath_itrtions, optmz_thrshld_min, optmz_thrshld_max, codepath_length, codepath_priorty y/o similares.

En una realización, la base de datos de DSFM puede interactuar con otros sistemas de base de datos. Por ejemplo, empleando un sistema de base de datos distribuido, accesos de consultas y datos mediante componente de DSFM de búsqueda pueden tratar la combinación de la base de datos de DSFM, una base de datos de capa de seguridad de datos integrada como una única entidad de base de datos.

En una realización, programas de usuario pueden contener diversas primitivas de interfaz de usuario, que pueden servir para actualizar el sistema de DSFM. También, diversas cuentas pueden requerir tablas de base de datos personalizadas que dependen de los entornos y los tipos de clientes que el sistema de DSFM puede necesitar servir. Se ha de observar que cualquier campo único puede designarse/etiquetarse como un campo clave en todo. En una realización alternativa, estas tablas se han descentralizado en sus propias bases de datos y sus respectivos controladores de base de datos (es decir, controladores de base de datos individuales para cada una de las anteriores tablas). Empleando técnicas de procesamiento de datos estándar, se puede distribuir adicionalmente las bases de datos a través de varias sistematizaciones informáticas y/o dispositivos de almacenamiento. De manera similar, configuraciones de controladores de base de datos descentralizada pueden variarse consolidando y/o distribuyendo los diversos componentes de base de datos 619a-f. El sistema de DSFM puede configurarse para controlar diversos ajustes, entradas y parámetros a través de controladores de base de datos.

La base de datos de DSFM puede comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. Más frecuentemente, la base de datos de DSFM comunica con uno o más componentes de DSFM, otros componentes de programa y/o similares. La base de datos puede contener, retener y proporcionar información con respecto a otros nodos y datos.

El componente de DSFM 635 es un componente de programa almacenado que se ejecuta por una CPU. En una realización, el componente de DSFM incorpora cualquiera y/o todas las combinaciones de los aspectos de las diversas realizaciones de un sistema de DSFM analizado en las figuras anteriores. Como tal, realizaciones del sistema de DSFM pueden afectar al acceso, obtención y la provisión de información, servicios, transacciones y/o similares a través de diversas redes de comunicaciones.

El uno o más componentes de DSFM pueden remediar una ejecución fallida en una instancia primaria de una aplicación de software en una ejecución de instancia secundaria de la aplicación de software que está en sinc con la ejecución de la instancia primaria antes del fallo de la misma. En una realización, componente de DSFM 635 toma entradas (por ejemplo, direcciones virtuales fragmentadas 204; y/o similares) etc., y transforma las entradas, a través de diversos componentes (por ejemplo, IMAS 641; y/o similares), en salidas (por ejemplo, espacio 206 de dirección virtual de memoria continua e infinita; y/o similar).

El componente o componentes de DSFM que habilitan acceso de información entre nodos pueden desarrollarse empleando herramientas de desarrollo estándar y lenguajes tales como, pero sin limitación: componentes de Apache, Ensamblador, ActiveX, ejecutables binarios, (ANSI) (Objective-) C (++), C# y/o.NET, adaptadores de bases de datos, guiones de CGI, Java, JavaScript, herramientas de correlación, herramientas de desarrollo orientadas a procedimientos y objetos, PERL, PHP, Python, guiones de intérprete de comandos, comandos SQL, extensiones de servidor de aplicación web, librerías y entornos de desarrollo web (por ejemplo, ActiveX de Microsoft; Adobe AIR, FLEX & FLASH; AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI); MooTools; Prototype; script.aculo.us; Protocolo Simple de Acceso a Objetos (SOAP); SWFObject; Interfaz de Usuario de Yahoo!; y/o similares), WebObjects y/o similares. En una realización, un servidor de DSFM emplea un servidor criptográfico para encriptar y desencriptar comunicaciones. El componente o componentes de DSFM pueden comunicar a y/o con otros componentes en una colección de componentes, incluyendo a sí mismo y/o instalaciones similares. Más frecuentemente, el componente o componentes de DSFM comunican con la base de datos de DSFM, sistemas operativos, otros componentes de programa y/o similares. Realizaciones del sistema de DSFM pueden contener, comunicar, generar, obtener y/o proporcionar componente de programa, sistema, usuario, y/o comunicaciones de datos, peticiones y/o respuestas.

La estructura y/u operación de cualquiera de los componentes de controlador de nodo de DSFM puede combinarse, consolidarse y/o distribuirse de cualquier número de formas para facilitar el desarrollo y/o despliegue. De manera similar, la colección de componentes puede combinarse de cualquier número de formas para facilitar el desarrollo

y/o despliegue. Para lograr esto, los componentes se pueden integrar en un código base común o en una instalación que puede cargar dinámicamente los componentes a demanda de una forma integrada.

La colección de componentes puede consolidarse y/o distribuirse en innumerables variaciones a través de procesamiento de datos estándar y/o técnicas de desarrollo. Múltiples instancias de uno cualquiera de los componentes de programa en la colección de componentes de programa pueden instanciarse en un único nodo, y/o a través de numerosos nodos para mejorar el rendimiento a través de equilibrio de carga y/o técnicas de procesamiento de datos. Adicionalmente, también pueden distribuirse instancias solas a través de múltiples controladores y/o dispositivos de almacenamiento; por ejemplo, bases de datos. Todas las instancias de componente de programa y controladores que trabajan de manera conjunta pueden hacerlo a través de técnicas de comunicación de procesamiento de datos estándar.

La configuración de las realizaciones de un controlador de DSFM puede depender del contexto de despliegue de sistema. Factores tales como, pero sin limitación, el presupuesto, capacidad, ubicación y/o uso de recursos de hardware subyacentes pueden afectar a requisitos y configuración de despliegue. Independientemente de si la configuración resulta en componentes de programa más consolidados y/o integrados, resulta en una serie más distribuida de componentes de programa, y/o resulta en alguna combinación entre una configuración consolidada y distribuida, datos pueden comunicarse, obtenerse y/o proporcionarse. Instancias de componentes consolidadas en un código base común de la colección de componentes de programa pueden comunicar, obtener y/o proporcionar datos. Esto puede conseguirse a través de técnicas de comunicación de procesamiento de datos intra aplicación tales como, pero sin limitación: referencia de datos (por ejemplo, apuntadores), mensajería interna, comunicación variable de instancia de objeto, espacio de memoria compartida, paso variable y/o similares.

Si los componentes de colección de componentes son discretos, separados y/o externos entre sí, entonces comunicar, obtener y/o proporcionar datos con y/o a otros componentes puede lograrse a través de técnicas de comunicación de procesamiento de datos inter aplicación tales como, pero sin limitación: paso de información de Interfaces de Programa de Aplicación (API); Modelo de Objeto de Componente (Distribuido) ((D)COM), Vinculación e Incorporación de Objetos (Distribuida) ((D)OLE) y/o similares), Arquitectura de Negociación de Petición de Objetos Comunes (CORBA), interfaces de programa de aplicación remotas y locales Jini, Notación de Objeto de JavaScript (JSON), Invocación de Método Remoto (RMI), SOAP, conductos de proceso, archivos compartidos y/o similares. Mensajes enviados entre componentes discretos para comunicación inter aplicación o dentro de espacios de memoria de un componente individual para comunicación intra aplicación pueden facilitarse a través de la creación y análisis de una gramática. Puede desarrollarse una gramática usando herramientas de desarrollo tales como lex, yacc, XML y/o similares, que permiten la generación de gramática y capacidades de análisis, que a su vez pueden formar la base de mensajes de comunicación dentro de y entre componentes.

Por ejemplo, puede disponerse una gramática para reconocer los testigos de un comando de publicación (*post*) de HTTP, por ejemplo:

```
w3c –post http://... Value1
```

donde Value1 se percibe como un parámetro porque "http://" es parte de la sintaxis de gramática, y lo que sigue se considera parte del valor de publicación. De manera similar, con una gramática de este tipo, una variable "Value1" puede insertarse en un comando de publicación de "http://" y enviarse a continuación. La propia sintaxis de gramática puede presentarse como datos estructurados que se interpretan y/o usan de otra manera para generar el mecanismo de análisis (por ejemplo, un archivo de texto de descripción de sintaxis como se procesa mediante lex, yacc, etc.). También, una vez que el mecanismo de análisis se genera y/o instancia, el propio mecanismo puede procesar y/o analizar datos estructurados tales como, pero sin limitación: texto delineado de carácter (por ejemplo, tabulación), HTML, flujos de texto estructurados, XML y/o datos estructurados similares. En otra realización, los propios protocolos de procesamiento de datos inter aplicación pueden tener analizadores integrados y/o fácilmente disponibles (por ejemplo, JSON, SOAP y/o analizadores similares) que pueden emplearse para analizar datos (por ejemplo, comunicaciones). Además, la gramática de análisis puede usarse más allá del análisis de mensajes, pero también puede usarse para analizar datos: bases de datos, colecciones de datos, almacenamientos de datos, datos estructurados y/o similares. De nuevo, la configuración deseada dependerá del contexto, entorno y requisitos de despliegue de sistema.

Por ejemplo, en algunas implementaciones, el controlador de DSFM puede estar ejecutando un guion de PHP que implementa un servidor de conexión de Capa de Conexiones Seguras ("SSL") a través del servidor de información, que escucha comunicaciones entrantes en un puerto de servidor al que un cliente puede enviar datos, por ejemplo, datos codificados en formato JSON. Tras identificar una comunicación entrante, un guion de PHP puede leer el mensaje entrante desde el dispositivo de cliente, analizar los datos de texto codificados por JSON recibidos para extraer información a partir de los datos de texto codificador por JSON en variables de guion de PHP y almacenar los datos (por ejemplo, información de identificación de cliente, etc.) y/o información extraída en una base de datos relacional accesible usando el Lenguaje de Consulta Estructurada ("SQL"). A continuación se proporciona un listado ilustrativo, escrito sustancialmente en forma de comandos de PHP/SQL, para aceptar datos de entrada codificados por JSON desde un dispositivo cliente a través de una conexión de SSL, analizar los datos para extraer variables y

almacenar los datos en una base de datos:

```
<?PHP
header('Content-Type: text/plain');

// establecer dirección IP y puerto para escuchar datos entrantes
$address = '192.168.0.100';
$port = 255;

// crear una conexión SSL de lado de servidor, escuchar/aceptar comunicación
entrante
$sock = socket_create(AF_INET, SOCK_STREAM, 0);
socket bind($sock, $address, $port) o die('No se pudo vincular a dirección');
socket listen($sock);
$client = socket accept($sock);

// leer datos de entrada de dispositivo cliente en bloques de 1024 byte hasta
final de mensaje do {
     $input =
     $input = socket_read($client, 1024);
     $data .= $input;
} while($input != "");

// analizar datos para extraer variables
$obj = json decode($data, true);

// almacenar datos de entrada en una base de datos
mysg1_connect("201.408.185.132",$DBserver,$password); // acceder a servidor de base
de datos
mysg1_select("CLIENT_DB.SQL"); // seleccionar base de datos a adjuntar
mysq1_query("INSERT INTO UserTable (transmission)
VALUES ($data)"); // añadir datos a tabla UserTable en una base de datos cliente
mysq1_close("CLIENT DB.SQL"); // cerrar conexión a base de datos
?>
```

También, los siguientes recursos pueden usarse para proporcionar realizaciones de ejemplo con respecto a implementación de analizador de SOAP:

```
http://www.xav.com/perl/site/lib/SOAP/Analizador.html
http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm
     .IBMDI.doc/referenceguide295.htm
```

y otras implementaciones de analizador:

```
http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm
     .IBMDI.doc/referenceguide259.htm
```

todos los cuales se incorporan expresamente por la presente mediante referencia en este documento.

Debe apreciarse que, dependiendo de las necesidades particulares y/o características de un DSFM individual y/o usuario empresarial, configuración de base de datos y/o modelo relacional, tipos de datos, transmisión de datos y/o marco de red, estructura de sintaxis y/o similares, diversas realizaciones del sistema de DSFM pueden implementarse que permiten un gran grado de flexibilidad y personalización. Mientras diversas realizaciones de un sistema de DSFM y descripciones de las mismas se han dirigido para optimización de ejecución de aplicación y eficacia, debe apreciarse que las realizaciones descritas en este documento pueden configurarse y/o personalizarse fácilmente para una amplia variedad de otras aplicaciones y/o implementaciones.

Mientras esta memoria descriptiva contiene muchos detalles de implementación específicos, estos no deberían interpretarse como limitaciones sobre el alcance de ninguna invención o de lo que puede reivindicarse, sino como descripciones de características específicas a realizaciones particulares de invenciones particulares. Ciertas características que se describen en esta memoria descriptiva en el contexto de realizaciones separadas también pueden implementarse en combinación en una única realización. A la inversa, diversas características que se describen en el contexto de una única realización también pueden implementarse en múltiples realizaciones de forma separada en cualquier subcombinación adecuada. Además, aunque características pueden describirse anteriormente como actuando en ciertas combinaciones e incluso inicialmente reivindicadas como tal, una o más características de una combinación reivindicada pueden eliminarse en algunos casos de la combinación, y la

combinación reivindicada puede dirigirse a una subcombinación o variación de una subcombinación.

De manera similar, aunque en los dibujos se representan operaciones en un orden particular, esto no debería entenderse como que se requiere que tales operaciones se realicen en el orden particular mostrado o en orden secuencial, o que se realicen todas las operaciones ilustradas, para conseguir resultados deseables. En ciertas circunstancias, la multitarea y el procesamiento paralelo pueden ser ventajosos. Además, la separación de diversos componentes de sistema en las realizaciones anteriormente descritas no debería entenderse como que se requiere tal separación en todas las realizaciones, y debería entenderse que los componentes de programa y sistemas descritos pueden integrarse generalmente juntos en un único producto de software o empaquetarse en múltiples productos de software.

Por lo tanto, se han descrito realizaciones particulares de la materia objeto. Otras realizaciones están dentro del alcance de las siguientes reivindicaciones. En algunos casos, las acciones citadas en las reivindicaciones pueden realizarse en un orden diferente y aún conseguir resultados deseables. Además, los procesos representados en las figuras adjuntas no requieren necesariamente el orden particular mostrado, u orden secuencial, para conseguir resultados deseables. En ciertas implementaciones, la multitarea y el procesamiento paralelo pueden ser ventajosos.

REIVINDICACIONES

1. Un método para ejecutar una aplicación de forma expeditiva en al menos un procesador informático, comprendiendo el método:

ejecutar simultáneamente una pluralidad de instancias de la aplicación en el al menos un procesador informático, esperándose que la aplicación produzca una secuencia de resultados, generándose cada instancia compilando el código fuente de la aplicación de acuerdo con una respectiva opción de compilador que es diferente de respectivas opciones de compilador usadas para generar todas las demás instancias de la pluralidad de instancias, en el que una primera instancia en la pluralidad de instancias produce salidas que corresponden a la secuencia de resultados, y cada una de las demás instancias en la pluralidad de instancias produce salidas que corresponden a la misma secuencia de resultados; y
para cada resultado de una primera secuencia de resultados a producir por la aplicación:

supervisar, correspondiendo al resultado, una respectiva salida generada por cada instancia; y
seleccionar, de las salidas supervisadas, la salida que ocurre primero como salida de la aplicación que corresponde al resultado, y etiquetar todas las demás salidas como duplicadas, acelerando de este modo un rendimiento informático del al menos un procesador informático.

2. El método de la reivindicación 1, en el que:

la pluralidad de instancias comprende una segunda instancia;
la opción de compilador para la primera instancia comprende compilación antes de tiempo (AIT); y
la opción de compilador para la segunda instancia comprende compilación justo a tiempo (JIT), en el que compilación JIT se basa en, al menos en parte, información de tiempo de ejecución obtenida a partir de al menos una ejecución anterior de la segunda instancia.

3. El método de la reivindicación 1, en el que:
la opción de compilador para la primera instancia se selecciona de un grupo que consiste en uso de memoria sin restricciones, minimización de uso de memoria, maximización de operaciones concurrentes y concurrencia restringida de operaciones.

4. El método de la reivindicación 1, comprendiendo adicionalmente:

etiquetar una instancia de la pluralidad de instancias como una instancia primaria;
etiquetar todas las demás instancias como instancias secundarias; y
para cada resultado de una segunda secuencia de resultados a producir por la aplicación, suprimir, correspondiendo al resultado, respectivas salidas de las instancias secundarias.

5. El método de la reivindicación 1, en el que al menos una porción de código fuente de la aplicación de software se especifica usando un lenguaje de programación que se interpreta al menos parcialmente.

6. Un sistema para ejecutar una aplicación de forma expeditiva en al menos un procesador informático, que comprende:

un primer procesador; y
una primera memoria en comunicación eléctrica con el primer procesador, comprendiendo la primera memoria instrucciones que, cuando se ejecutan por una unidad de procesamiento que comprenden al menos uno de un primer procesador y un segundo procesador, programan la unidad de procesamiento para:

ejecutar simultáneamente una pluralidad de instancias de la aplicación en el al menos un procesador informático, esperándose que la aplicación produzca una secuencia de resultados, generándose cada instancia compilando el código fuente de la aplicación de acuerdo con una respectiva opción de compilador que es diferente de respectivas opciones de compilador usadas para generar todas las demás instancias de la pluralidad de instancias, en el que una primera instancia en la pluralidad de instancias produce salidas que corresponden a la secuencia de resultados, y cada una de las demás instancias en la pluralidad de instancias produce salidas que corresponden a la misma secuencia de resultados; y para cada resultado de una primera secuencia de resultados a producir por la aplicación:

supervisar, correspondiendo al resultado, una respectiva salida generada por cada instancia; y
seleccionar, de las salidas supervisadas, la salida que ocurre primero como salida de la aplicación que corresponde al resultado, y etiquetar todas las demás salidas como duplicadas, acelerando de este modo un rendimiento informático del al menos un procesador informático.

7. El sistema de la reivindicación 6, en el que:

la pluralidad de instancias comprende una segunda instancia;
la opción de compilador para la primera instancia comprende compilación antes de tiempo (AIT); y
la opción de compilador para la segunda instancia comprende compilación justo a tiempo (JIT), en el que compilación JIT se basa en, al menos en parte, información de tiempo de ejecución obtenida a partir de al menos una ejecución anterior de la segunda instancia.

8. El sistema de la reivindicación 6, en el que:
la opción de compilador para la primera instancia se selecciona de un grupo que consiste en uso de memoria sin restricciones, minimización de uso de memoria, maximización de operaciones concurrentes y concurrencia restringida de operaciones.

9. El sistema de la reivindicación 6, en el que las instrucciones adicionalmente programan la unidad de procesamiento para:

etiquetar una instancia de la pluralidad de instancias como una instancia primaria;
etiquetar todas las demás instancias como instancias secundarias; y
para cada resultado de una segunda secuencia de resultados a producir por la aplicación, suprimir, correspondiendo al resultado, respectivas salidas de las instancias secundarias.

10. El sistema de la reivindicación 6, en el que al menos una porción de código fuente de la aplicación de software se especifica usando un lenguaje de programación que se interpreta al menos parcialmente.

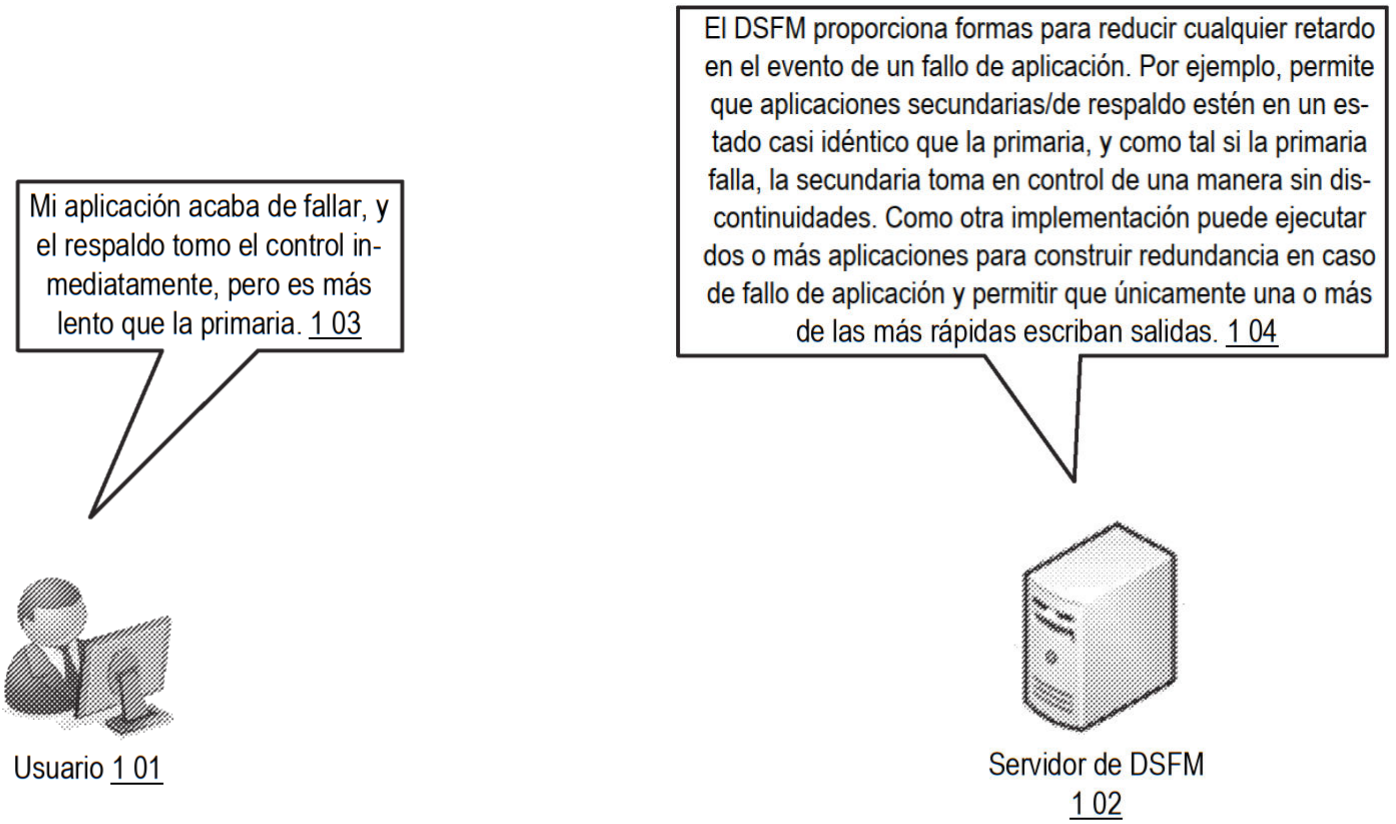

Figura 1 Implementación de ejemplo: gestión de optimización, sincronización y conmutación por error de aplicación

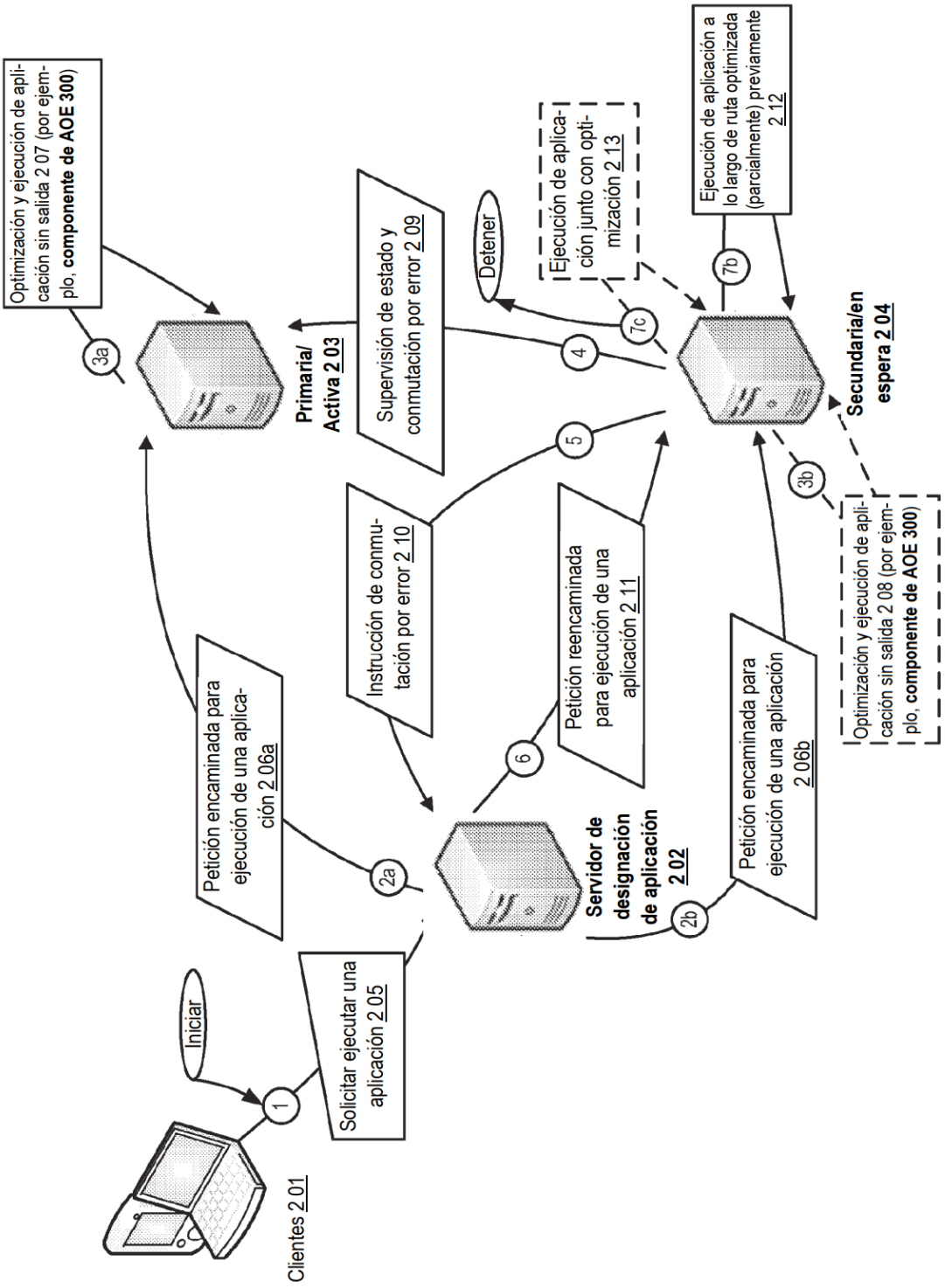

Figura 2

Implementación de ejemplo: optimización de aplicación, sincronización y procedimiento de conmutación por error

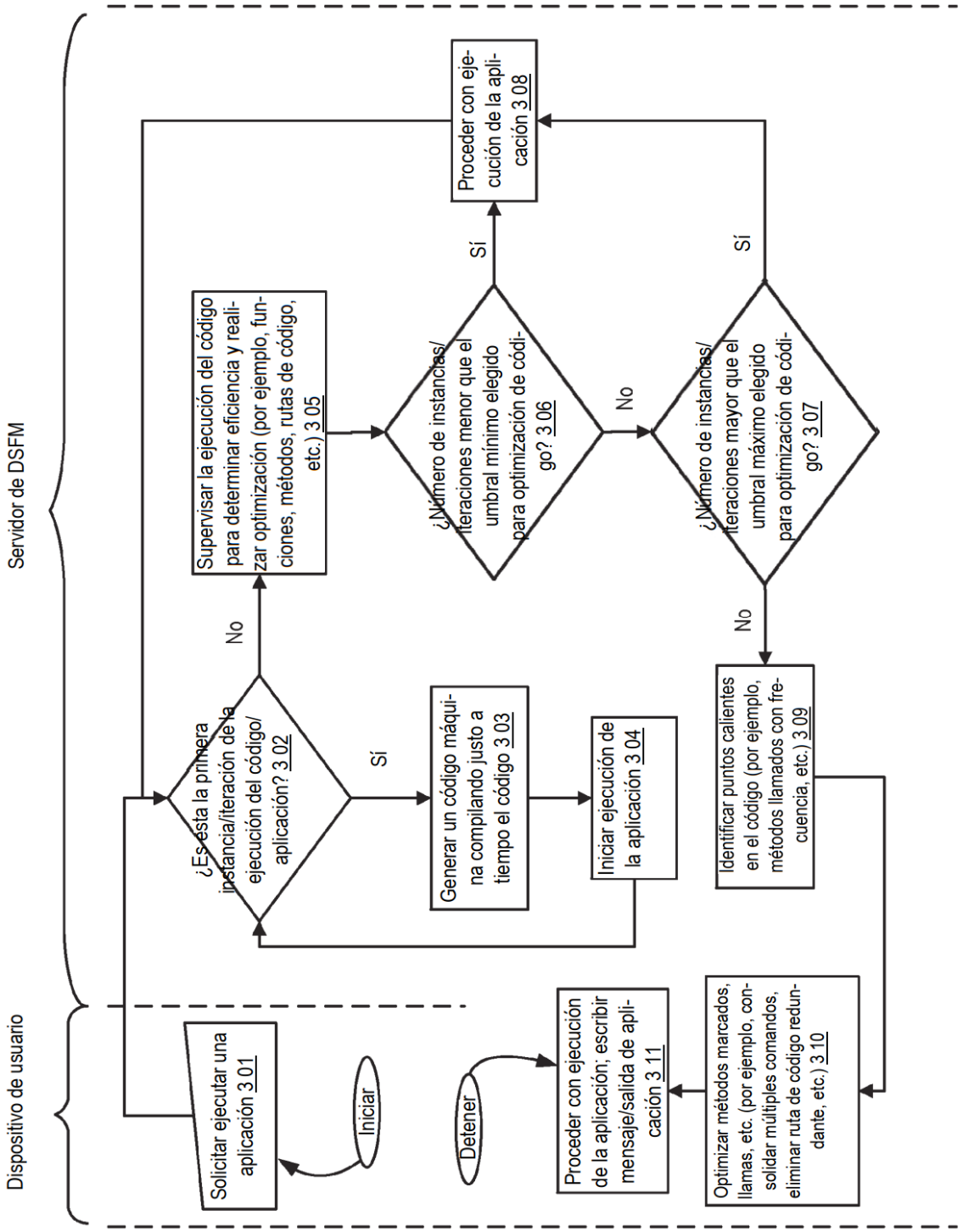

Figura 3 Implementación de ejemplo: componente de ejecución y optimización de aplicación (AOE)

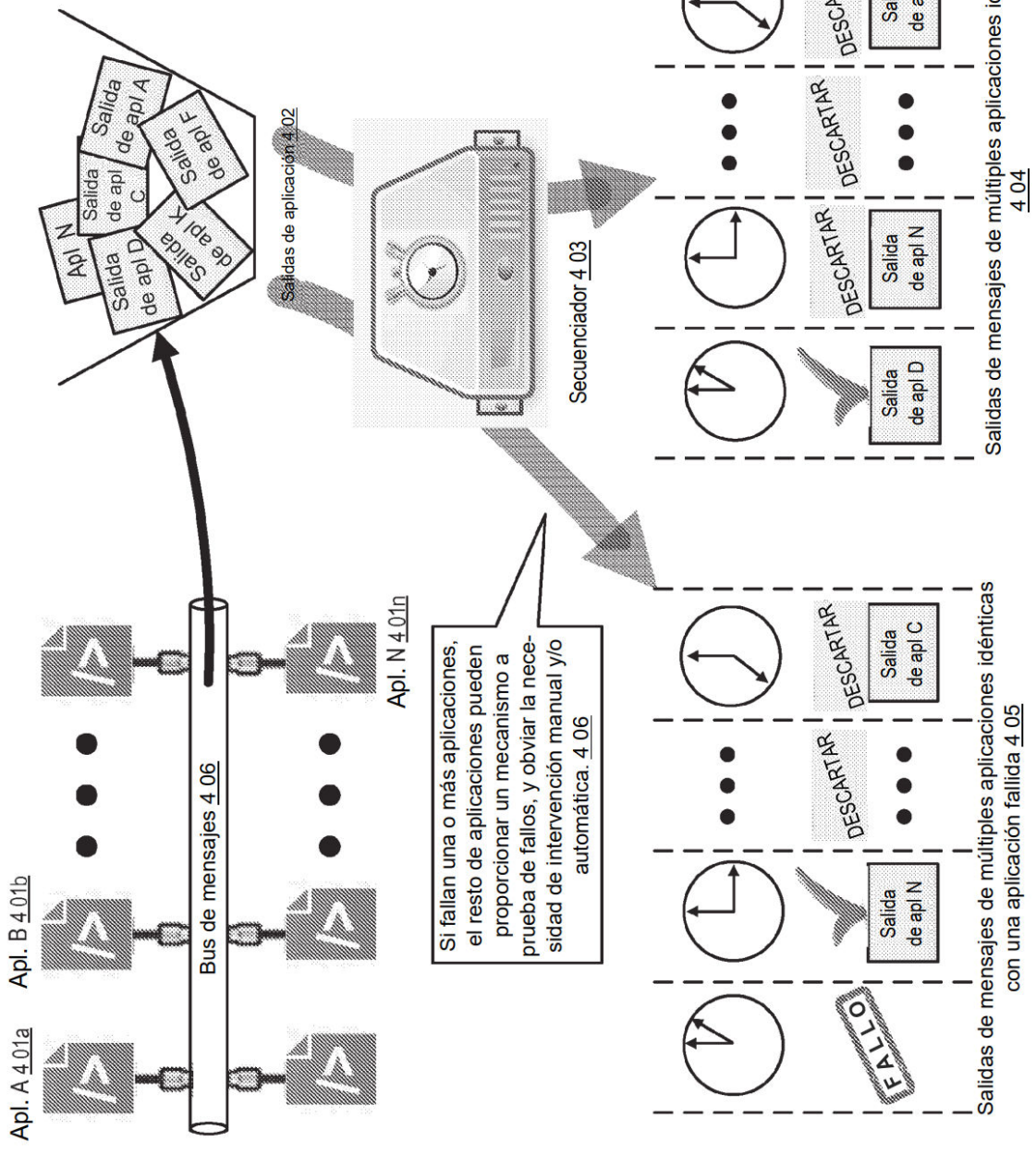

FIG. 5

Figura 4

Implementación de ejemplo: sincronización de conmutación por error a través de procedimiento de redundancia

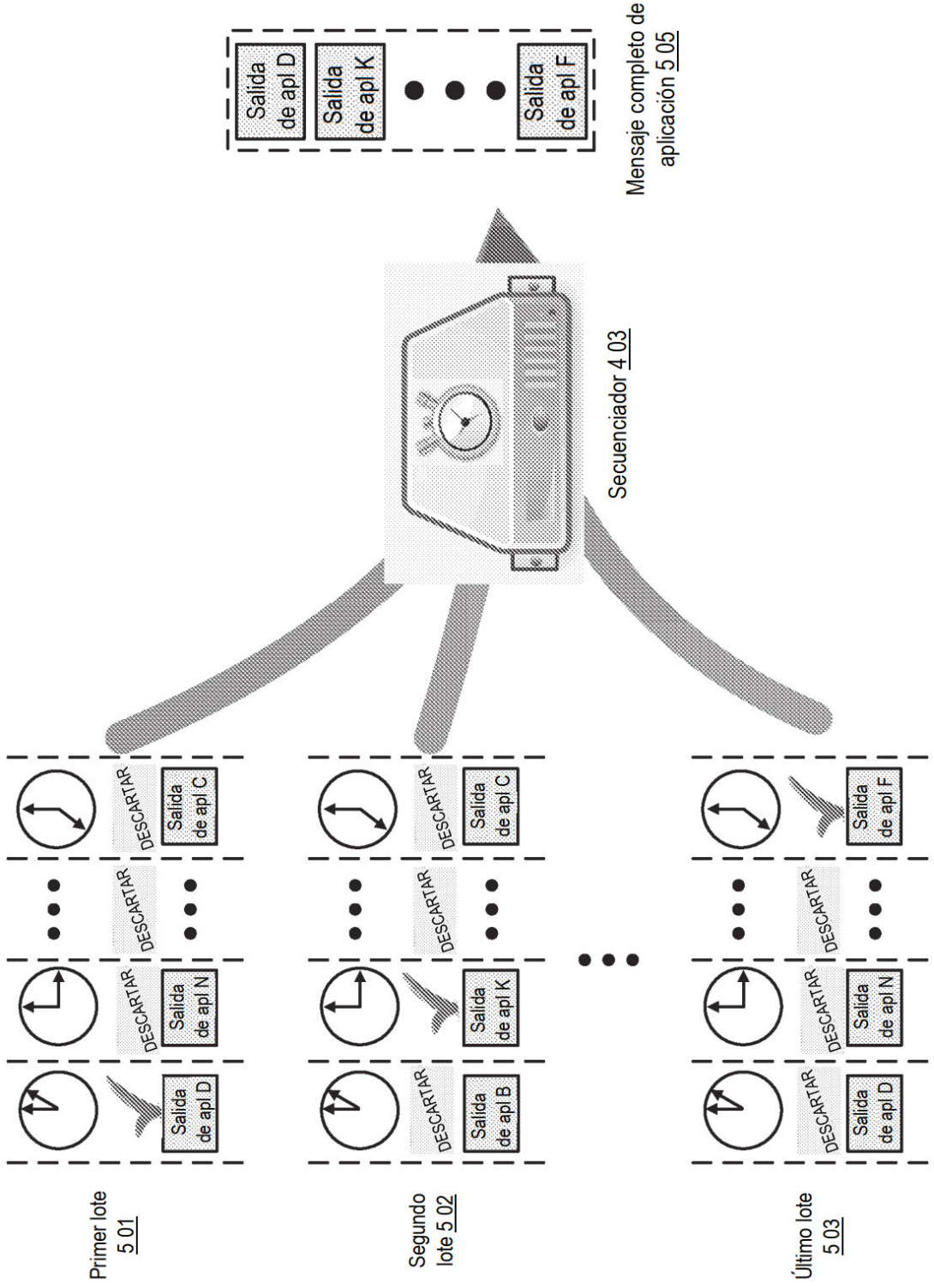

Figura 5

Implementación de ejemplo: sincronización de conmutación por error a través de procedimiento de redundancia

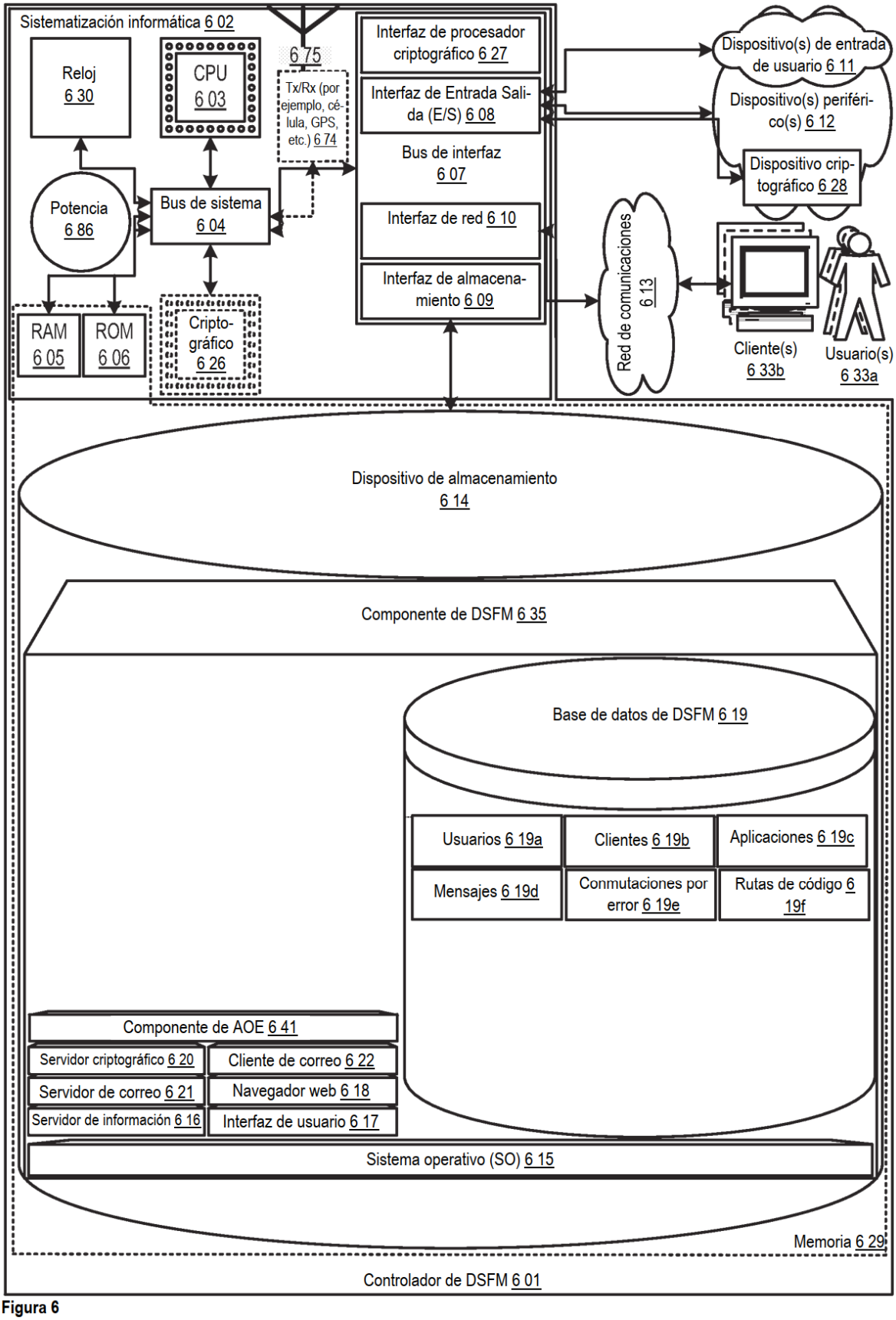
Sistematización informática 6 02
Reloj 6 30
CPU 6 03
6 75
Tx/Rx (por ejemplo, célula, GPS, etc.) 6 74
Potencia 6 86
Bus de sistema 6 04
RAM 6 05
ROM 6 06
Cripto-gráfico 6 26
Interfaz de procesador criptográfico 6 27
Interfaz de Entrada Salida (E/S) 6 08
Bus de interfaz 6 07
Interfaz de red 6 10
Interfaz de almacenamiento 6 09
Dispositivo(s) de entrada de usuario 6 11
Dispositivo(s) periféricos 6 12
Dispositivo criptográfico 6 28
Red de comunicaciones 6 13
Cliente(s) 6 33b
Usuario(s) 6 33a
Dispositivo de almacenamiento 6 14
Componente de DSFM 6 35
Base de datos de DSFM 6 19
Usuarios 6 19a
Clientes 6 19b
Aplicaciones 6 19c
Mensajes 6 19d
Conmutaciones por error 6 19e
Rutas de código 6 19f
Componente de AOE 6 41
Servidor criptográfico 6 20
Cliente de correo 6 22
Servidor de correo 6 21
Navegador web 6 18
Servidor de información 6 16
Interfaz de usuario 6 17
Sistema operativo (SO) 6 15
Memoria 6 29
Controlador de DSFM 6 01

Figura 6