(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 645 414**

(51) Int. CI.:

G06F 11/14 (2006.01)
G06F 11/20 (2006.01)
G06F 17/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **26.10.2010 PCT/US2010/054139**

(87) Fecha y número de publicación internacional: **05.05.2011 WO11053594**

(96) Fecha de presentación y número de la solicitud europea: **26.10.2010 E 10827394 (7)**

(97) Fecha y número de publicación de la concesión europea: **27.09.2017 EP 2494444**

(54) Título: **Conmutación por error y recuperación para instancias de datos replicados**

(30) Prioridad:

26.10.2009 US 606097

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
05.12.2017

(73) Titular/es:

AMAZON TECHNOLOGIES, INC. (100.0%)
P.O. Box 8102
Reno, NV 89507, US

(72) Inventor/es:

MCALISTER, GRANT ALEXANDER MACDONALD
y
SIVASUBRAMANIAN, SWAMINATHAN

(74) Agente/Representante:

PONS ARIÑO, Ángel

DESCRIPCIÓN

Conmutación por error y recuperación para instancias de datos replicados

ANTECEDENTES

A medida que aumenta el número de aplicaciones y servicios que están disponibles a través de redes como Internet, un número creciente de contenido, aplicaciones y/o proveedores de servicios recurren a tecnologías tales como la computación en nube. La computación en nube, en general, es un enfoque para proporcionar acceso a recursos electrónicos a través de servicios, tales como servicios Web, donde el hardware y/o el software utilizado para soportar esos servicios es dinámicamente ampliable para satisfacer las necesidades de los servicios en un momento dado. Típicamente, un usuario o cliente alquilará, arrendará o pagará de otro modo por el acceso a recursos a través de la nube y, por lo tanto, no tendrá que comprar y mantener el hardware y/o software para proporcionar acceso a estos recursos.

Aunque los aspectos de diversas aplicaciones y recursos pueden ajustarse y gestionarse en la nube, los repositorios de datos sobre los cuales dependen estas aplicaciones y recursos no son ajustables de forma similar ni pueden ser fácilmente gestionados por un cliente u otro usuario. Típicamente, realizar tareas como aprovisionar y ampliar el almacenamiento de datos son procedimientos manuales tediosos, en los que un cliente tiene que proporcionar un administrador de base de datos (DBA) o un usuario experto similar con información y requisitos de configuración, de tal forma que el DBA pueda determinar si la configuración es válida. Además, no existe una forma fácil para un cliente de ajustar de forma dinámica y/o automática los parámetros de una instancia de base de datos o gestionar otros aspectos de un repositorio de datos. En muchos casos, una instancia de datos tendrá mecanismos de respaldo y recuperación en su lugar, pero estos mecanismos a menudo están en una única ubicación o área de tal manera que son susceptibles a fallos o interrupciones en esa área. Además, cuando una instancia de datos falla, típicamente suele tardar algunos minutos en generar una nueva instancia, adjuntar los volúmenes adecuados a la nueva instancia y, de otro modo, realizar las tareas necesarias para recuperarse del error.

El documento US 2007/198700 A1 desvela un sistema de base de datos configurado para la conmutación por error automática con pérdida de datos limitada de usuario. En la configuración automática de conmutación por error, un sistema de base de datos primario permanece disponible incluso en ausencia de un modo de espera y de un Observador, siempre y cuando el modo de espera y el Observador estén ausentes secuencialmente. Hay dos maneras básicas para que el Observador decida que se debe intentar una conmutación por error. Uno de ellos es que el Observador determina, a partir del fallo del primario al responder al ping del Observador, que el primario está ausente. Si el ping que el Observador envía al servidor primario no recibe una respuesta a tiempo, el Observador volverá a intentar el ping de acuerdo con un algoritmo de reintento que factores en el valor de un umbral. Si no se puede establecer contacto con el primario antes de que expire el umbral, el Observador asume que la base de datos primaria ha fallado y comienza la secuencia de conmutación por error. Otra forma es que el servidor primario informe al Observador de que es necesaria una conmutación por error. La función en el servidor primario que responde al ping del Observador realizará ciertas comprobaciones internas para determinar si existen condiciones que requieran una conmutación por error. Un valor de encarnación de metadatos (MIV) es un número cada vez mayor que se aumenta para cada cambio que se hace alguna vez en los metadatos DG. Se desvela además un procedimiento para la propagación de una configuración de conmutación por error automática (AFC). El estado AFC se propaga mediante mensajes que contienen copias del estado AFC. Estos mensajes acompañan a los pings.

Por lo tanto, el objeto de la presente invención es proporcionar un método mejorado de gestión de una instancia de base de datos replicada en un entorno de base de datos utilizando un entorno de control separado, así como el sistema correspondiente.

Este objeto se resuelve por la materia objeto de las reivindicaciones independientes 1 y 10.

Las realizaciones preferidas se definen por las reivindicaciones dependientes.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Se describirán diversas realizaciones de acuerdo con la presente divulgación con referencia a los dibujos, en los que:

La FIG. 1 ilustra un entorno en el que pueden implementarse diversas realizaciones;

la FIG. 2 ilustra una separación ejemplar de un plano de control y un plano de datos que se pueden usar de acuerdo con diversas realizaciones;

la FIG. 3 ilustra un ejemplo que utiliza una pluralidad de componentes de supervisión que se pueden usar de acuerdo con diversas realizaciones;

la FIG. 4 ilustra una implementación ejemplar para ejecutar una instancia de datos replicada a través de múltiples zonas de datos que se pueden usar de acuerdo con una realización;

la FIG. 5 ilustra un diagrama de transición de estado ejemplar para una réplica primaria de acuerdo con una realización;

la FIG. 6 ilustra un diagrama de transición de estado ejemplar para un componente de supervisión de acuerdo con una realización;

la FIG. 7 ilustra un proceso ejemplar para realizar una operación de conmutación por error que puede utilizarse de acuerdo con una realización;

la FIG. 8 ilustra un proceso ejemplar para recuperar una réplica secundaria que se puede usar de acuerdo con una realización;

la FIG. 9 ilustra un proceso ejemplar para gestionar procesadores de eventos que se puede usar de acuerdo con una realización;

la FIG. 10 ilustra un ejemplo de una reasignación debido a un procesador de eventos fallidos que se puede usar de acuerdo con una realización; y

la FIG. 11 ilustra un proceso ejemplar para añadir un nuevo procesador de eventos que se puede usar de acuerdo con una realización.

DESCRIPCIÓN DETALLADA

Los sistemas y métodos de acuerdo con diversas realizaciones de la presente divulgación pueden superar una o más de las mencionadas anteriormente y otras deficiencias experimentadas en enfoques convencionales para gestionar aspectos del almacenamiento de datos en un entorno electrónico. En particular, diversas realizaciones proporcionan un entorno de control separado, o un plano de control, que puede usarse para permitir que un usuario administre y/o altere diversos aspectos de un entorno de datos o plano de datos. Esta funcionalidad de "autoservicio" puede proporcionarse a través de un conjunto de servicios Web, permitiendo que el usuario y el plano de control actúen juntos como un administrador de base de datos virtual (DBA). Un usuario o cliente puede enviar una solicitud al plano de control a través de una de una pluralidad de interfaces de programación de aplicaciones (API) visibles externamente, por ejemplo. Se pueden usar varias API para realizar funciones específicas con respecto a un repositorio de datos, tal como una base de datos relacional, en el entorno de datos. Una solicitud recibida a una de las API puede analizarse para determinar la acción o acciones deseadas a realizar en el plano de datos, tales como acciones que ajustan parámetros operativos o de configuración de un almacén de datos o instancia de almacenamiento de datos. Un componente tal como un componente de flujo de trabajo puede determinar las tareas apropiadas para la acción, y hacer que las tareas se ejecuten en un orden apropiado. Al menos una de estas tareas se realizará típicamente en el entorno de datos, tal como para ajustar un aspecto de una base de datos relacional.

De acuerdo con ciertas realizaciones, tal sistema puede proporcionar el aprovisionamiento de una instancia de datos replicados en el entorno de datos. El aprovisionamiento puede utilizar un enfoque de replicación primaria-secundaria, estando cada una de las réplicas primarias y secundarias aprovisionadas en o entre una o más zonas de datos separadas, ubicaciones geográficas separadas, etc. Las réplicas de base de datos pueden ejecutarse en instancias de datos independientes, cada una unida a volúmenes de almacenamiento de bloques dedicados que no se comparten entre las réplicas.

En diversas realizaciones, la replicación puede realizarse utilizando un mecanismo de replicación a nivel de bloque, tal como un Dispositivo de Bloque Replicado Distribuido (DRBD®) de Linbit de Viena, Austria, o un

almacén de bloques elástico (EBS), según se proporciona por Amazon.com, Inc., de Seattle, Washington, que puede reflejar el contenido de dispositivos de bloque entre servidores y replicar de forma síncrona datos a través de sistemas redundantes. Cada instancia puede ejecutar un núcleo que tiene instalado un módulo de núcleo de mecanismo de replicación a nivel de bloques (BLRM) instalado para gestionar todas las operaciones de entrada y salida (I/O) para la instancia de datos. Todas las lecturas y escrituras se pueden ejecutar en una réplica primaria, replicando el mecanismo de replicación a nivel de bloques la información de forma síncrona con la réplica secundaria.

Tanto las réplicas primarias como las secundarias pueden tener un nombre DNS externo. Los clientes pueden alcanzar la réplica primaria actual con un nombre DNS tal como DNS_primary. El nombre DNS_primary puede ser un alias o "cname" con respecto al nombre DNS externo de la réplica primaria (actual). Cuando una réplica primaria falla o no está disponible, la réplica secundaria puede promocionarse o fallar para convertirse en la nueva réplica primaria, por lo que el cname para DNS_primary puede actualizarse al nombre DNS de la nueva instancia primaria. Todas las escrituras se envían a la base de datos de la réplica primaria actual. Cuando la instancia primaria recibe una escritura, la información se escribe de forma síncrona en la réplica secundaria. Después de escribir con éxito en ambos lugares, la escritura se puede considerar exitosa. Todas las lecturas también se ejecutan sólo en la réplica primaria en diversas realizaciones.

Por lo tanto, la replicación de bases de datos puede ser soportada en múltiples instancias de datos utilizando réplicas de instancia que se ejecutan en diferentes zonas de datos. Las escrituras de base de datos se pueden validar utilizando un mecanismo de replicación síncrono a nivel de bloques, de tal forma que no se pierden datos a menos que ninguna de las réplicas esté disponible debido a una interrupción a gran escala que implique múltiples zonas de datos, etc. La replicación puede proporcionar una mayor disponibilidad que la que puede conseguirse usando una única instancia de base de datos, ya que un único fallo de réplica no provoca una interrupción en la base de datos durante un periodo de tiempo prolongado. Por ejemplo, si la réplica primaria de una base de datos está inactiva, diversas realizaciones pueden realizar una operación de conmutación por error por lo que una réplica secundaria toma el relevo como la nueva réplica primaria. La replicación también puede proporcionar una durabilidad mayor que una base de datos no replicada en muchos casos, protegiendo contra fallos de una zona de datos, fallos de volumen de datos, etc.

La FIG. 1 ilustra un ejemplo de un entorno 100 para implementar aspectos de acuerdo con diversas realizaciones. Como se apreciará, aunque se utiliza un entorno basado en Web con fines de explicación, pueden utilizarse diferentes entornos, según sea apropiado, para implementar diversas realizaciones. El entorno 100 mostrado incluye tanto una porción (o lado) de prueba o desarrollo como una porción de producción. La porción de producción incluye un dispositivo de cliente electrónico 102, que puede incluir cualquier dispositivo apropiado operable para enviar y recibir peticiones, mensajes o información a través de una red apropiada 104 y transmitir información de nuevo a un usuario del dispositivo. Los ejemplos de tales dispositivos de cliente incluyen ordenadores personales, teléfonos móviles, dispositivos de mensajería de mano, ordenadores portátiles, decodificadores, asistentes de datos personales, lectores de libros electrónicos y similares. La red puede incluir cualquier red apropiada, incluyendo una intranet, Internet, una red celular, una red de área local o cualquier otra red o combinación de las mismas. Los componentes utilizados para tal sistema pueden depender, al menos en parte, del tipo de red y/o entorno seleccionado. Los protocolos y componentes para la comunicación a través de dicha red son bien conocidos y no se analizarán en el presente documento en detalle. La comunicación a través de la red se puede habilitar mediante conexiones cableadas o inalámbricas, y combinaciones de las mismas. En este ejemplo, la red incluye Internet, ya que el entorno incluye un servidor Web 106 para recibir solicitudes y servir contenido en respuesta a las mismas, aunque para otras redes podría usarse un dispositivo alternativo que sirva a un propósito similar como será evidente para uno experto en la técnica.

El entorno ilustrativo incluye al menos un servidor de aplicaciones 108 y un almacén de datos 110. Debe entenderse que puede haber varios servidores de aplicaciones, capas u otros elementos, procesos o componentes, que pueden estar encadenados o configurados de otro modo, que pueden interactuar para realizar tareas como obtener datos de un almacén de datos adecuado. Como se usa en el presente documento, la expresión "almacén de datos" se refiere a cualquier dispositivo o combinación de dispositivos capaz de almacenar, acceder y recuperar datos, que puede incluir cualquier combinación y número de servidores de datos, bases de datos, dispositivos de almacenamiento de datos y medios de almacenamiento de datos, en cualquier entorno estándar, distribuido o agrupado. El servidor de aplicaciones puede incluir cualquier hardware y software apropiado para integrarse con el almacén de datos según sea necesario para ejecutar aspectos de una o más aplicaciones para el dispositivo de cliente, manejando la mayoría del acceso de datos y lógica de negocio para una aplicación. El servidor de aplicaciones proporciona servicios de control de acceso en cooperación con el almacén de datos y es capaz de generar contenido como texto, gráficos, audio y/o vídeo para ser transferido al

usuario, que puede ser servido al usuario por el servidor Web en forma de HTML, XML u otro lenguaje estructurado apropiado en este ejemplo. La manipulación de todas las solicitudes y respuestas, así como la entrega de contenido entre el dispositivo de cliente 102 y el servidor de aplicaciones 108, puede ser manejada por el servidor Web. Debe entenderse que los servidores Web y de aplicaciones no son necesarios y son simplemente componentes ejemplares, ya que el código estructurado analizado en el presente documento se puede ejecutar en cualquier dispositivo apropiado o máquina anfitriona, como se analiza en otra parte en el presente documento. Además, el entorno se puede diseñar de tal manera que se pueda proporcionar un marco de automatización de prueba como un servicio al que un usuario o aplicación puede suscribirse. Se puede proporcionar un marco de automatización de prueba como una implementación de cualquiera de los diversos patrones de prueba analizados en el presente documento, aunque también se pueden utilizar diversas implementaciones diferentes, como se analiza o sugiere en el presente documento.

El entorno también incluye un lado de desarrollo y/o prueba, que incluye un dispositivo de usuario 118 que permite a un usuario, tal como un desarrollador, administrador de datos o probador acceder al sistema. El dispositivo de usuario 118 puede ser cualquier dispositivo o máquina apropiados, tal como se ha descrito anteriormente con respecto al dispositivo de cliente 102. El entorno también incluye un servidor de desarrollo 120, que funciona similar al servidor de aplicaciones 108, pero que típicamente ejecuta código durante el desarrollo y la prueba antes de que el código se implemente y se ejecute en el lado de producción y sea accesible a usuarios externos, por ejemplo. En algunas realizaciones, un servidor de aplicaciones puede funcionar como un servidor de desarrollo, y no se puede usar almacenamiento separado de producción y prueba.

El almacén de datos 110 puede incluir varias tablas de datos separadas, bases de datos u otros mecanismos y medios de almacenamiento de datos para almacenar datos relacionados con un aspecto particular. Por ejemplo, el almacén de datos ilustrado incluye mecanismos para almacenar datos de producción 112 e información de usuario 116, que pueden usarse para servir contenido para el lado de producción. También se muestra que el almacén de datos incluye un mecanismo para almacenar datos de prueba 114, que pueden usarse con la información de usuario para el lado de prueba. Debe entenderse que puede haber muchos otros aspectos que pueden necesitar ser almacenados en el almacén de datos, tales como información de imagen de página e información de derecho de acceso, que pueden almacenarse en cualquiera de los mecanismos enumerados anteriormente, según sea apropiado o en mecanismos adicionales en el almacén de datos 110. El almacén de datos 110 es operable, a través de la lógica asociada con el mismo, para recibir instrucciones del servidor de aplicaciones 108 o servidor de desarrollo 120, y obtener, actualizar o procesar datos de otro modo en respuesta a los mismos. En un ejemplo, un usuario puede enviar una solicitud de búsqueda para un determinado tipo de elemento. En este caso, el almacén de datos puede acceder a la información de usuario para verificar la identidad del usuario, y puede acceder a la información de detalle del catálogo para obtener información sobre los elementos de ese tipo. Después, la información puede devolverse al usuario, tal como en un listado de resultados en una página Web que el usuario puede ver a través de un navegador en el dispositivo de usuario 102. La información para un artículo particular de interés se puede ver en una página dedicada o ventana del navegador.

Cada servidor incluirá típicamente un sistema operativo que proporcione instrucciones de programa ejecutables para la administración general y operación de ese servidor, e incluirá típicamente un medio legible por ordenador que almacena instrucciones que, al ejecutarse por un procesador del servidor, permiten que el servidor realice sus funciones previstas. Las implementaciones adecuadas para el sistema operativo y la funcionalidad general de los servidores se conocen o están disponibles en el mercado, y se implementan fácilmente por expertos en la técnica, particularmente a la luz de la divulgación en el presente documento.

El entorno en una realización es un entorno informático distribuido que utiliza varios sistemas y componentes informáticos que están interconectados a través de enlaces de comunicación, utilizando una o más redes informáticas o conexiones directas. Sin embargo, los expertos en la técnica apreciarán que tal sistema podría funcionar igualmente bien en un sistema que tenga menos o un mayor número de componentes que los que se ilustran en la Fig. 1. Por lo tanto, la representación del sistema 100 en la Fig. 1 debe considerarse de carácter ilustrativo y no limitarse al alcance de la divulgación.

Un entorno tal como el ilustrado en la Fig. 1 puede ser útil para un proveedor, tal como un mercado electrónico, en el que se pueden usar múltiples anfitriones para realizar tareas tales como servir contenido, autenticar usuarios, realizar transacciones de pago, o realizar cualquiera de otras tareas. Algunos de estos anfitriones pueden configurarse para ofrecer la misma funcionalidad, mientras que otros servidores pueden estar configurados para realizar al menos algunas funciones diferentes. El entorno electrónico en tales casos podría incluir componentes adicionales y/u otras disposiciones, tales como las ilustradas en la configuración 200 de la

Fig. 2, que se analiza en detalle a continuación.

Los sistemas y métodos de acuerdo con una realización proporcionan un servicio de base de datos relacional ("RDS") que permite a desarrolladores, clientes u otros usuarios autorizados de manera fácil y rentable obtener y configurar bases de datos relacionales y otras fuentes de datos para que los usuarios puedan realizar tareas tales como almacenar, procesar y consultar conjuntos de datos relacionales en una nube. Si bien este ejemplo se analiza con respecto a Internet, servicios Web y tecnología basada en Internet, debe entenderse que los aspectos de las diversas realizaciones pueden usarse con cualquier servicio apropiado disponible u ofrecido a través de una red en un entorno electrónico. Además, aunque el servicio se denomina en el presente documento como un "servicio de base de datos relacional", debe entenderse que dicho servicio puede utilizarse con cualquier tipo apropiado de repositorio de datos o almacenamiento de datos en un entorno electrónico. Un RDS en este ejemplo incluye al menos un servicio Web que permite a los usuarios o clientes administrar fácilmente conjuntos de datos relacionales sin preocuparse por las complejidades administrativas de implementación, actualizaciones, gestión de parches, copias de seguridad, replicación, conmutación por error, gestión de capacidad, escalado, y otros aspectos diferentes de la gestión de datos. Por lo tanto, los desarrolladores se liberan para desarrollar sofisticadas aplicaciones en la nube sin preocuparse de las complejidades de la gestión de la infraestructura de la base de datos.

Un RDS en una realización proporciona un "plano de control" separado que incluye componentes (por ejemplo, hardware y software) útiles para gestionar aspectos del almacenamiento de datos. En una realización, se proporciona un conjunto de interfaces de programación de aplicaciones (API) de gestión de datos u otras interfaces diferentes que permiten a un usuario o cliente hacer llamadas en el RDS para realizar ciertas tareas relacionadas con el almacenamiento de datos. El usuario todavía puede utilizar las interfaces directas o las API para comunicarse con los repositorios de datos, sin embargo, y puede utilizar las API específicas de RDS del plano de control sólo cuando sea necesario para gestionar el almacenamiento de datos o realizar una tarea similar.

La FIG. 2 ilustra un ejemplo de una implementación RDS 200 que se puede usar de acuerdo con una realización. En este ejemplo, se muestra un dispositivo informático 202 para un usuario final capaz de hacer llamadas a través de una red 206 en un plano de control 208 para realizar una tarea tal como proporcionar un repositorio de datos del plano de datos 210. El usuario o una aplicación 204 puede acceder al repositorio proporcionado directamente a través de una interfaz de un plano de datos 210. Aunque se utiliza un dispositivo informático de usuario final y una aplicación para fines de explicación, debe entenderse que cualquier usuario, aplicación, servicio, dispositivo, componente o recurso puede acceder a la interfaz o interfaces del plano de control y/o plano de datos según sea apropiado en las diversas realizaciones. Además, aunque los componentes están separados en "planos" de control y datos, debe entenderse que esto puede referirse a una separación real o virtual de al menos algunos recursos (por ejemplo, hardware y/o software) utilizados para proporcionar la funcionalidad respectiva.

El plano de control 208 en este ejemplo es esencialmente una capa virtual de componentes de hardware y software que maneja acciones de control y gestión, tales como aprovisionamiento, ampliación, replicación, etc. El plano de control en esta realización incluye una capa de servicios Web 212 o nivel, que puede incluir al menos un servidor Web, por ejemplo, junto con software ejecutable por ordenador, servidores de aplicaciones u otros componentes de este tipo. La capa de servicios Web también puede incluir un conjunto de API 232 (u otras interfaces de este tipo) para recibir llamadas o solicitudes de servicios Web de la red 206. Cada API puede proporcionarse para recibir solicitudes de al menos una acción específica que se debe realizar con respecto al entorno de datos, como proporcionar, ampliar, clonar o hibernar una instancia de una base de datos relacional. Tras recibir una solicitud a una de las API, la capa de servicios Web puede analizar o examinar la solicitud para determinar las etapas o acciones necesarias para actuar o procesar la llamada. Por ejemplo, se puede recibir una llamada de servicio Web que incluye una solicitud para crear un repositorio de datos. En este ejemplo, la capa de servicios Web puede analizar la solicitud para determinar el tipo de repositorio de datos a crear, el volumen de almacenamiento solicitado, el tipo de hardware solicitado (si existe), u otros aspectos de este tipo. La información para la solicitud se puede escribir en un almacén de datos de administración ("Admin") 222, u otra ubicación de almacenamiento apropiada o cola de trabajo, para su posterior procesamiento.

Una capa de servicio Web en una realización incluye un conjunto escalable de servidores orientados al cliente que pueden proporcionar las diversas API de plano de control y devolver las respuestas apropiadas basadas en las especificaciones de la API. La capa de servicio Web también puede incluir al menos una capa de servicio de API que en una realización consiste en servidores apilados y replicados que procesan las API de cliente orientadas hacia el exterior. La capa de servicios Web puede ser responsable de las características frontales del

servicio Web, tal como autenticar clientes basándose en credenciales, autorizar al cliente, limitar las solicitudes de los clientes a los servidores de API, validar la entrada de usuarios, y serializar o deserializar solicitudes y respuestas. La capa de API también puede ser responsable de leer y escribir datos de configuración de base de datos a/desde el almacén de datos de administración, en respuesta a las llamadas de la API. En muchas realizaciones, la capa de servicios Web y/o capa de servicio API será el único componente visible externamente, o el único componente visible y accesible para los clientes del servicio de control. Los servidores de la capa de servicios Web pueden no tener estado y escalarse horizontalmente como se conoce en la técnica. Los servidores API, así como el almacén de datos persistente, se pueden distribuir a través de múltiples centros de datos en una región geográfica, o cerca de una ubicación geográfica, por ejemplo, de tal forma que los servidores sean resistentes a fallos de un único centro de datos.

El plano de control en esta realización incluye lo que se denomina en el presente documento como un componente "rastreador" 214. Un componente rastreador puede ser cualquier componente apropiado operable para hacer sondeo de diversos componentes del plano de control o determinar de otro modo cualquier tarea a ejecutar en respuesta a una solicitud pendiente. En este ejemplo, la capa de servicios web puede poner instrucciones o información para la solicitud "crear base de datos" en el almacén de datos de administración 222 o una cola de trabajo similar, y el rastreador puede comprobar periódicamente el almacén de datos de administración para trabajos pendientes. Pueden usarse otros diversos enfoques como será evidente para un experto en la técnica, tal como la capa de servicios Web que envía una notificación a un rastreador de que existe un trabajo. El componente rastreador puede capturar la solicitud de "crear base de datos", y usando información para la solicitud puede enviar una solicitud, llamada u otro comando similar a un componente de flujo de trabajo 216 operable para instanciar al menos un flujo de trabajo para la solicitud. El flujo de trabajo en una realización se genera y se mantiene usando un servicio de flujo de trabajo como se analiza en otra parte en el presente documento. Un flujo de trabajo en general es una secuencia de tareas que se deben ejecutar para realizar un trabajo específico. El flujo de trabajo no es el trabajo real, sino una abstracción del trabajo que controla el flujo de información y ejecución del trabajo. Un flujo de trabajo también puede considerarse como una máquina de estado, que puede gestionar y devolver el estado de un proceso en cualquier momento durante la ejecución. Un componente de flujo de trabajo (o sistema de componentes) en una realización es operable para administrar y/o realizar el alojamiento y ejecución de flujos de trabajo para tareas tales como: creación, modificación y eliminación de repositorios; recuperación y respaldo; creación, eliminación y modificación de grupos de seguridad; gestión de credenciales de usuario; y la rotación de claves y la gestión de credenciales. Dichos flujos de trabajo se pueden implementar en la parte superior de un servicio de flujo de trabajo, como se analiza en otra parte en el presente documento. El componente de flujo de trabajo también puede gestionar diferencias entre las etapas de flujo de trabajo utilizadas para diferentes motores de base de datos, tal como MySQL, ya que el servicio de flujo de trabajo subyacente no cambia necesariamente.

En este ejemplo, se puede instanciar un flujo de trabajo utilizando una plantilla de flujo de trabajo para crear una base de datos y aplicar información extraída de la solicitud original. Por ejemplo, si la solicitud es para una instancia de sistema de gestión de base de datos relacional (RDBMS) MySQL®, a diferencia de un RDBMS de Oracle® u otra instancia de este tipo, se añadirá entonces una tarea específica al flujo de trabajo que se dirige a instancias de MySQL. El componente de flujo de trabajo también puede seleccionar tareas específicas relacionadas con la cantidad de almacenamiento solicitada, los requisitos de hardware específicos, u otras tareas similares. Estas tareas se pueden añadir al flujo de trabajo en un orden de ejecución útil para el trabajo en general. Mientras que algunas tareas se pueden realizar en paralelo, otras tareas dependen de las tareas anteriores a completarse en primer lugar. El componente o servicio de flujo de trabajo puede incluir esta información en el flujo de trabajo, y las tareas se pueden ejecutar y pasarse información según sea necesario.

Un ejemplo de flujo de trabajo de "crear base de datos" para un cliente puede incluir tareas tales como aprovisionar una instancia de almacén de datos, asignar un volumen de almacenamiento persistente fuera de instancia, adjuntar el volumen de almacenamiento persistente a la instancia de almacén de datos, después asignar y adjuntar una dirección DNS o otra dirección, puerto, interfaz o identificador que el cliente pueda utilizar para acceder o conectarse de otro modo a la instancia de datos. En este ejemplo, se proporciona a un usuario la dirección DNS y una dirección de puerto para utilizarla para acceder a la instancia. El flujo de trabajo también puede incluir tareas para descargar e instalar cualquier información binaria u otra información utilizada para la tecnología de almacenamiento de datos específica (por ejemplo, MySQL). El componente de flujo de trabajo puede gestionar la ejecución de éstas y cualquier tarea relacionada, o cualquier otra combinación apropiada de dichas tareas, y puede generar una respuesta a la solicitud que indique la creación de una "base de datos" en respuesta a la solicitud "crear base de datos", que corresponde realmente a una instancia de almacén de datos en el plano de datos 210, y proporcionar la dirección DNS que se utilizará para acceder a la instancia. Después, un usuario puede acceder a la instancia del almacén de datos directamente usando la dirección y el puerto DNS,

sin tener que acceder o pasar a través del plano de control 208. Pueden usarse diversas plantillas de flujo de trabajo diferentes para realizar trabajos similares, tales como suprimir, crear o modificar una de más instancias de almacén de datos, tal como para aumentar el almacenamiento. En algunas realizaciones, la información de flujo de trabajo se escribe en almacenamiento, y al menos un componente de ejecución separado (no mostrado) extrae o de otro modo, accede o recibe tareas a ejecutar basándose en la información de flujo de trabajo. Por ejemplo, puede haber un componente de aprovisionamiento dedicado que ejecute tareas de aprovisionamiento, y este componente no puede ser llamado por el componente de flujo de trabajo, pero puede supervisar una cola de tareas o puede recibir información para una tarea de aprovisionamiento en cualquiera de varias maneras relacionadas como debe ser aparente.

Como se ha mencionado, diversas realizaciones pueden aprovechar un servicio de flujo de trabajo que puede recibir peticiones o llamadas para un estado actual de un proceso o tarea, tal como el aprovisionamiento de un repositorio, y puede devolver el estado actual del proceso. El componente de flujo de trabajo y/o el servicio de flujo de trabajo no realizan las llamadas o solicitudes reales para realizar cada tarea, sino que gestionan la información de estado y configuración para el flujo de trabajo que permite a los componentes del plano de control determinar la siguiente tarea que se va a realizar; y cualquier información necesaria para esa tarea, a continuación, generar la llamada o llamadas adecuadas en el plano de datos que incluye esa información de estado, por lo que un componente del plano de datos puede hacer la llamada para realizar la tarea. Los flujos de trabajo y las tareas se pueden programar en paralelo con el fin de aumentar el rendimiento y maximizar los recursos de procesamiento. Como se analiza, la realización real de las tareas se producirá en el plano de datos, pero las tareas se originarán desde el plano de control. Por ejemplo, el componente de flujo de trabajo puede comunicarse con un administrador de anfitrión, que puede realizar llamadas al almacén de datos. Por lo tanto, para una tarea dada se podría hacer una llamada al servicio de flujo de trabajo pasando ciertos parámetros, por lo que el servicio de flujo de trabajo genera la secuencia de tareas para el flujo de trabajo y proporciona el estado actual, de modo que se puede realizar una tarea para el estado actual. Una vez realizada la tarea (o se resuelve o se concluye de otra manera), un componente, tal como el gestor anfitrión, puede responder al servicio, que puede proporcionar información sobre el siguiente estado en el flujo de trabajo, de tal forma que se pueda realizar la siguiente tarea. Cada vez que se realiza una de las tareas para el flujo de trabajo, el servicio puede proporcionar una nueva tarea que se realizará hasta que se complete el flujo de trabajo. Además, se pueden ejecutar múltiples subprocesos en paralelo para diferentes flujos de trabajo con el fin de acelerar el procesamiento del flujo de trabajo.

El plano de control 208 en esta realización también incluye al menos un componente de supervisión 218. Cuando se crea una instancia de datos en el plano de datos, la información para la instancia puede escribirse en un almacén de datos en el plano de control, tal como un almacén de datos de supervisión 220. Debe entenderse que el almacén de datos de supervisión puede ser un almacén de datos separado, o puede ser una porción de otro almacén de datos tal como un conjunto distinto de tablas en un almacén de datos de Admin 222, u otro repositorio apropiado. Un componente de supervisión puede acceder a la información del almacén de datos de supervisión para determinar las instancias activas 234 en el plano de datos 210. Un componente de supervisión también puede realizar otras tareas, tales como recopilar información de registro y/o eventos de múltiples componentes del plano de control y/o el plano de datos, tal como la capa de servicios Web, el componente de flujo de trabajo, el componente rastreador, y diversos administradores de anfitriones. Al utilizar dicha información de eventos, el componente de supervisión puede exponer eventos visibles para el cliente, con el fin de implementar API orientadas al cliente. Un componente de supervisión puede supervisar constantemente el estado de todos los repositorios y/o instancias en ejecución para el plano de control, detectar el fallo de cualquiera de estas instancias, e iniciar el proceso o procesos de recuperación apropiados.

Cada instancia 234 en el plano de datos puede incluir al menos un almacén de datos 226 y un componente de administrador de anfitrión 228 para la máquina que proporciona acceso al almacén de datos. Un administrador de anfitrión en una realización es una aplicación o agente de software que se ejecuta en una instancia y/o un servidor de aplicaciones, tal como un servidor de aplicaciones Tomcat o Java, programado para gestionar tareas tales como implementación de software y operaciones de almacén de datos, así como supervisar n estado del almacén de datos y/o la instancia respectiva. Un administrador de anfitrión en una realización escucha en un puerto que sólo puede ser alcanzado desde los componentes internos del sistema, y no está disponible para clientes u otras entidades externas. En algunas realizaciones, el administrador de anfitrión no puede iniciar ninguna llamada en la capa de plano de control. Un administrador de anfitrión puede ser responsable de administrar y/o realizar tareas tales como configurar las instancias de un nuevo repositorio, incluyendo la configuración de volúmenes lógicos y sistemas de archivos, la instalación de binarios de bases de datos y semillas, y el inicio o detención del repositorio. Un administrador de anfitrión puede supervisar el estado del almacén de datos, así como supervisar el almacén de datos para determinar condiciones de error, tales como

errores de I/O o errores de almacenamiento de datos, y puede reiniciar el almacén de datos si es necesario. Un administrador de anfitrión también realiza y/o gestiona la instalación de parches y actualizaciones de software para el almacén de datos y/o el sistema operativo. Un administrador de anfitrión también puede recopilar métricas relevantes, tal como las relacionadas con la CPU, la memoria y el uso de I/O.

El componente de supervisión puede comunicarse periódicamente con cada administrador de anfitrión 228 para las instancias supervisadas 234, tal como enviando una solicitud específica o supervisando las pulsaciones de los administradores de anfitriones, para determinar el estado de cada anfitrión. En una realización, el componente de supervisión incluye un conjunto de procesadores de eventos (o servidores de supervisión) configurados para emitir comandos a cada administrador de anfitrión, tales como para obtener el estado de un anfitrión y/o instancia particular. Si no se recibe una respuesta después de un número especificado de reintentos, entonces el componente de supervisión puede determinar que existe un problema y puede almacenar información en el almacén de datos de Admin 222 u otra cola de trabajos diferente para realizar una acción para la instancia, tal como para verificar el problema y volver a proporcionar la instancia si es necesario. El rastreador puede acceder a esta información e iniciar un flujo de trabajo de recuperación para que la instancia intente recuperarse automáticamente del fallo. El administrador de anfitrión 228 puede actuar como un proxy para la supervisión y otros componentes del plano de control, realizando tareas para las instancias en nombre de los componentes del plano de control. De vez en cuando, se producirá un problema con una de las instancias, tal como la caída del anfitrión, instancia o volumen correspondiente, el reinicio, la reactivación, etc., que no pueden resolverse automáticamente. En una realización, hay un componente de registro (no mostrado) que puede registrar estos y otros eventos de visibilidad del cliente. El componente de registro puede incluir una API u otra interfaz de este tipo de tal forma que si una instancia no está disponible durante un periodo de tiempo, un cliente puede llamar a un "evento" apropiado o API similar para obtener la información relacionada con el evento. En algunos casos, una solicitud puede dejarse pendiente cuando falla una instancia. Dado que el plano de control en esta realización está separado del plano de datos, el plano de control nunca recibe la solicitud de datos y, por lo tanto, no puede poner en cola la solicitud para una presentación posterior (aunque en algunas realizaciones, esta información podría ser enviada al plano de control). Por lo tanto, el plano de control en esta realización proporciona información al usuario con respecto al fallo para que el usuario pueda manejar la solicitud según sea necesario.

Como se analiza, una vez que se ha proporcionado una instancia y se proporciona a un usuario una dirección DNS u otra dirección o ubicación, el usuario puede enviar solicitudes "directamente" al plano de datos 210 a través de la red utilizando una conectividad de base de datos Java (JDBC) u otro cliente diferente para interactuar directamente con dicha instancia 234. En una realización, el plano de datos adopta la forma de (o al menos incluye o forma parte de) un entorno de computación en la nube, o un conjunto de servicios y recursos Web que proporciona almacenamiento de datos y acceso a través de una red en la "nube" o dinámica de componentes de hardware y/o software. Una dirección DNS es beneficiosa en tal entorno de nube dinámico, ya que los fallos de instancia o disponibilidad, por ejemplo, pueden enmascararse correlacionando de nuevo programaticamente una dirección DNS con cualquier instancia de reemplazo apropiada para un uso. Una solicitud recibida de un usuario 202 o una aplicación 204, por ejemplo, puede dirigirse a un enrutador de traducción de dirección de red (NAT) 224 u otro componente apropiado, que puede dirigir la solicitud a la instancia real 234 o anfitrión correspondiente al DNS de la solicitud. Como se analiza, tal enfoque permite que las instancias se muevan dinámicamente, se actualicen, se repliquen, etc., sin que el usuario o la aplicación deba cambiar el DNS u otra dirección utilizada para acceder a la instancia. Como se analiza, cada instancia 234 puede incluir un administrador de anfitrión 228 y un almacén de datos 226 y puede tener al menos una instancia de copia de seguridad o copia en almacenamiento permanente 230. Utilizando tal enfoque, una vez que la instancia ha sido configurada a través del plano de control, un usuario, aplicación, servicio o componente puede interactuar con la instancia directamente a través de las solicitudes al plano de datos, sin tener que acceder al plano de control 232. Por ejemplo, el usuario puede emitir directamente lenguaje de consulta estructurado (SQL) u otros comandos relacionados con los datos en la instancia a través de la dirección DNS. El usuario sólo tendrá que acceder al plano de control si el usuario desea realizar una tarea tal como expandir la capacidad de almacenamiento de una instancia. En al menos una realización, la funcionalidad del plano de control 208 puede ofrecerse como al menos un servicio por un proveedor que puede o no estar relacionado con un proveedor del plano de datos 210, pero que puede ser simplemente un servicio de terceros que puede utilizarse para proporcionar y gestionar instancias de datos en el plano de datos, y también puede supervisar y asegurar la disponibilidad de estas instancias en un plano de datos separado 210.

Como se analiza, una ventaja de proporcionar la funcionalidad de un plano de control como un servicio Web u otro servicio similar es que el plano de control funciona como un administrador de base de datos virtual (DBA) y evita la necesidad de un DBA humano para realizar tareas tales como aprovisionamiento de datos. El

aprovisionamiento de datos es actualmente un tedioso procedimiento manual, que requiere que un DBA reciba la información de configuración necesaria, determine si la configuración es válida, optimice y sintonice la instancia, y realice otras tareas de este tipo, que requieren mucho tiempo y esfuerzo. Además, tal enfoque proporciona muchas oportunidades para el error, lo que podría no ser descubierto hasta después de que se pierden los datos. Utilizando un plano o servicio de control como se describe en el presente documento, un usuario o cliente en cambio puede enviar una llamada que incluye información tal como un tipo de hardware y una versión de un producto de base de datos. El plano o servicio de control puede realizar entonces las tareas necesarias para crear, eliminar, modificar, expandir o modificar de otra forma un almacén de datos o una instancia de almacenamiento de datos. El plano de control también puede soportar varios motores de base de datos diferentes de una manera consistente, sin necesidad de que un DBA sea un experto en cada uno de los motores. Una vez aprovisionado, el usuario tiene acceso nativo a la instancia o instancias de datos, y simplemente puede dirigir las aplicaciones existentes (tales como aplicaciones MySQL) a la dirección DNS u otra información de ubicación para la instancia particular. No existe ninguna restricción o modificación de los modelos de consulta u otra funcionalidad de este tipo, ya que un usuario puede seguir utilizando aplicaciones basadas en MySQL, Oracle, u otra tecnología de base de datos.

La FIG. 3 ilustra un ejemplo de una configuración 300 que puede usarse para fines tales como la supervisión y recuperación automatizada de instancias de RDS, simples o replicadas, de acuerdo con una realización. Aunque los números de referencia se transfieren entre las figuras con fines de simplicidad y claridad, debe entenderse que éstas representan meramente componentes similares que pueden utilizarse para diversas realizaciones, y no debe interpretarse que requieran componentes de diversas realizaciones diferentes o que muestren simplemente diferentes vistas de una única realización. Además, pueden utilizarse menos componentes o componentes adicionales en diversas realizaciones, y la presencia o ausencia de un componente en una figura dada no debe interpretarse como ese componente se requiera o no sea útil en una realización dada a menos que se indique otra cosa específicamente. Las variaciones entre las realizaciones y las figuras deben ser evidentes para un experto en la materia a la luz de la presente divulgación.

Como se ilustra en la figura, un componente (o servicio) de supervisión 218 del plano de control puede comprender una serie de nodos de procesamiento 302, denominados en el presente documento como procesadores de eventos. En una realización, los procesadores de eventos comprenden una flota de servidores de supervisión operables para supervisar aspectos del plano de datos. Cada procesador de eventos puede configurarse para comunicarse con un conjunto o intervalo especificado de almacenes de datos 226 y/o instancias de datos 234 a través del administrador de anfitrión asociado 228. Como se analiza, cada almacén de datos y administrador de anfitrión puede existir en un nodo o máquina del plano de datos 210, o entorno de datos. Cada uno de los procesadores de eventos puede comunicarse con los administradores de anfitriones asignados utilizando cualquier técnica de comunicación apropiada para obtener un estado actual de cada anfitrión, por ejemplo, haciendo ping a cada administrador de anfitrión utilizando una solicitud segura (por ejemplo, HTTPS), tal como una solicitud "getStatus". En respuesta a la petición, cada administrador de anfitrión puede enviar una respuesta que incluye información tal como si hay un problema con, o detectada por, el administrador de anfitrión 228, así como cualquier métrica, valores de parámetro, o información de diagnóstico relevante que se determine relevante. En ciertas realizaciones, la cantidad y el tipo de información devuelta por un administrador de anfitrión pueden variar en función de un estado del administrador de anfitrión. Por ejemplo, si no se detectan errores, el administrador de anfitrión puede enviar un conjunto estándar de métricas especificadas a registrar o procesar de otro modo. Si se detecta un problema, por ejemplo, puede incluirse un conjunto de información diferente, tal como información que indica el tipo de problema, así como información de diagnóstico u otra información relevante para ese tipo de problema. Pueden proporcionarse varios algoritmos a los administradores de anfitriones para realizar tales determinaciones. Tras recibir la información de los administradores de anfitriones, los procesadores de eventos pueden analizar la información, según sea necesario, y almacenar la información en un almacén de datos de supervisión 220 u otra ubicación similar. Los procesadores de eventos también pueden almacenar cualquier información de registro, que se analiza en otra parte en el presente documento, en el almacén de datos de supervisión. Como se ilustra en este ejemplo, el almacén de datos de supervisión 220 puede ser un único almacén de datos lógicos, pero puede dividirse en muchas instancias de datos 304.

Puede haber muchas ventajas en el uso de múltiples procesadores de eventos 302 como parte del componente de supervisión 218. Una ventaja de este tipo es que, para un gran número de instancias de datos 234 en el plano de datos, un procesador de eventos único puede no tener suficiente capacidad para supervisar cada instancia al mismo tiempo. La utilización de múltiples procesadores de eventos permite que el trabajo de supervisión se distribuya a través de varios procesadores de eventos. Además, el uso de múltiples procesadores de eventos permite a los procesadores de eventos existentes asumir el trabajo de otro procesador de eventos en el caso de

un fallo u otro problema similar. Si una instancia de datos sólo se gestionó por un único procesador de eventos, y hubo un problema con el procesador haciendo que el procesador de eventos no estuviese disponible, entonces esa instancia de datos podría no tener ninguna supervisión realizada, por lo tanto, podría arriesgarse a una interrupción u otro problema de este tipo. Al extender la supervisión a través de un conjunto de procesadores de eventos, y permitir que el intervalo de supervisión por cada procesador de eventos se actualice dinámicamente, el plano de control puede asegurar que cada instancia en el plano de datos se supervisa sustancialmente en cualquier momento, incluso en el caso de un fallo de uno o más de los procesadores de eventos.

En una realización, la responsabilidad de cada procesador de eventos se determina tomando el número de instancias (incluidas las réplicas) a supervisar en cualquier momento dado y proporcionando el número de instancias a través del número de procesadores de eventos. Por ejemplo, si hay 25.000 instancias a supervisar en el plano de datos, y hay cinco procesadores de eventos en ejecución en el plano de control, cada procesador de eventos puede tener la responsabilidad de supervisar aproximadamente 5.000 de las instancias de datos. Si a cada instancia se le da un identificador, por ejemplo, entonces cada procesador de eventos puede recibir una serie de identificadores (tal como los primeros 5.000 identificadores, los segundos 5.000 identificadores, etc.) para facilitar la adaptación de la responsabilidad de cada procesador de eventos, en lugar de tener que administrar la información de correlación para cada una de las 25.000 instancias. El ejemplo de la figura muestra la gama de responsabilidades para cada uno de los procesadores de eventos en un ejemplo de este tipo.

En un intervalo apropiado, tal como una vez por minuto, cada procesador de eventos 302 puede enviar una solicitud a cada administrador de anfitrión 228 que está siendo supervisado por ese procesador de eventos. Un procesador de eventos en una realización es una aplicación Java que se ejecuta en un contenedor Tomcat del plano de control que encuesta periódicamente a los administradores de anfitriones para instancias de datos en el plano de datos. El procesador de eventos puede encuestar un administrador de anfitrión en una realización haciendo una llamada getStatus( ) o similar (por ejemplo, por SSL) usando el nombre DNS y el puerto de administrador de anfitrión. En algunas realizaciones, una instancia de datos que se está supervisando se identifica únicamente mediante una combinación de un identificador de almacén de datos de cliente, un identificador de almacén de datos, y un identificador de instancia. Utilizando tal enfoque, se pueden distinguir los estados de las instancias viejas y nuevas al mover una instancia de datos a otra instancia en la nube. El procesador de eventos puede determinar el estado de la instancia de datos basándose en la respuesta del administrador de anfitrión. Una instancia de datos en una realización puede estar en uno de al menos los siguientes estados ejemplares: "OK" (la instancia de datos se está ejecutando correctamente), "incomunicada" (la instancia de datos está en un estado sospechoso de fallo), o "muerta" (la instancia de datos es inaccesible y no responde a solicitudes de estado).

En la mayoría de los casos, el administrador de anfitrión devolverá una respuesta indicando que el administrador de anfitrión, la instancia asociada, etc., se está ejecutando como se esperaba, y el procesador de eventos puede actualizar la información en el almacén de datos de supervisión 220. Un procesador de eventos puede considerar que una instancia de datos está en un estado "OK" o similar en una realización cuando el administrador de anfitrión devuelve una respuesta apropiada, tal como un código de respuesta HTTP "200" (un código de respuesta estándar para solicitudes HTTP satisfactorias). Si no se recibe una respuesta de un administrador de anfitrión, o si la respuesta es una respuesta de tiempo de espera (tal como el código HTTP "500", o cualquier otro código de respuesta de error "5xx"), el procesador de eventos puede volver a enviar la solicitud getStatus, y puede poner la instancia de base de datos en un estado "incomunicado" o similar. Si el anfitrión ha estado en estado "incomunicado" durante más de un número predeterminado de pings de estado, u otras solicitudes de ese tipo, entonces se puede declarar que la instancia de datos está en estado "inactivo" o similar. Si el anfitrión vuelve a estar en línea con un código de respuesta "200" (o similar) dentro del número predeterminado de pings de estado, el anfitrión o instancia se puede mover a un estado "OK". El número predeterminado de comprobaciones antes de mover un estado de anfitrión de "incomunicado" a "inactivo" u "OK" utilizado, al menos en parte, sirve para evitar falsos positivos debido a errores de red intermitentes, procesadores de eventos temporalmente sobrecargados, administradores de anfitriones temporalmente sobrecargados, u otros errores temporales de este tipo que en realidad no dan como resultado una instancia de datos no disponible que de otro modo requiere recuperación. En una realización, no se mantiene un estado de "incomunicado", ya que el estado puede determinarse fácilmente por otro procesador de eventos.

Si no se recibe una respuesta después del número predeterminado de solicitudes de estado, o si el estado se mueve de otro modo a un estado "inactivo" o similar, como se ha analizado en otra parte del presente documento, el procesador de eventos introduce información relativa al estado del problema en el almacén de datos de Admin 222 (u otra cola de trabajo diferente como se ha analizado anteriormente) que indica que hay un estado sospechoso con respecto al administrador de anfitrión que no responde. Como se ha analizado

anteriormente, un componente rastreador 214 del plano de control puede comprobar periódicamente el almacén de datos de Admin para determinar información, y cuando el rastreador detecta la información para el estado sospechoso o problemático, se puede iniciar un flujo de trabajo de recuperación apropiado. Por ejemplo, el rastreador puede pasar información al componente de flujo de trabajo 216 que hace que se genere un flujo de trabajo adecuado, tal como un flujo de trabajo para manejar una instancia de datos que no esté disponible, un flujo de trabajo para manejar errores notificados por un administrador de anfitrión, o cualquiera de un número de otras situaciones de este tipo. El administrador de flujo de trabajo puede generar el flujo de trabajo apropiado, pasar información de estado y manejar diversos aspectos diferentes como se analiza en otra parte del presente documento.

Una ventaja de almacenar la información de recuperación en el almacén de datos de Admin es que tal enfoque permite la recuperación incluso en el caso de un fallo del sistema de supervisión. Puede ser deseable permitir acciones de recuperación independientes de la disponibilidad del almacén de datos de supervisión. Puede ser aceptable utilizar el almacén de datos de Admin, ya que en esta realización cualquier tipo de recuperación, incluyendo la generación de un flujo de trabajo, etc., requiere que el almacén de datos de Admin (u otra cola de trabajo diferente) está activo y disponible. Por lo tanto, puede ser deseable evitar colocar otra dependencia en la recuperación, y en lugar de tener un único lugar de disponibilidad.

Los sistemas y métodos de acuerdo con diversas realizaciones permiten a los clientes utilizar servicios Web, o un enfoque similar de este tipo, para crear una o más instancias de base de datos replicadas en un entorno de computación en la nube o similar, proporcionando una solución de datos altamente duradera y altamente disponible. Cuando un cliente crea una instancia de base de datos replicada en diversas realizaciones, los datos de cliente se replican de forma síncrona utilizando un modelo de replicación primario-secundario. En algunas realizaciones, las réplicas pueden estar localizadas en diferentes ubicaciones físicas, tales como en diferentes zonas de datos. Cada "zona" de datos puede referirse a uno o más centros de datos, o grupos de servidores de datos, por ejemplo, situados dentro de un área geográfica específica, localizándose zonas diferentes en o alrededor de diferentes ubicaciones geográficas. Una instancia RDS puede tolerar entonces el fallo de una de las zonas de datos, ya que otra zona de datos en una ubicación geográfica diferente puede evitar el fallo, excepto en el caso de un evento catastrófico grande. En algunos casos, un centro de datos puede abarcar múltiples zonas de datos, pero las réplicas de datos dentro de un centro de datos dado pueden ser instanciadas en diferentes zonas. Son posibles muchas otras variaciones, tales como zonas de superposición, zonas en múltiples ubicaciones geográficas, etc. Si una réplica primaria falla o de otro modo, deja de estar disponible, el sistema RDS puede conmutar por error rápida y automáticamente a la réplica secundaria, dando como resultado muy poco tiempo de inactividad o indisponibilidad de datos.

En una realización, un cliente es capaz de crear una instancia de base de datos replicada llamando a una interfaz especificada de la capa de servicios Web del plano de control, tal como se analiza con respecto a la Fig. 2. Por ejemplo, un cliente puede llamar a una API "CreateDBInstance" especificando aspectos tales como la clase de instancia, el almacenamiento asignado, el motor de la base de datos, etc., ya que el cliente creará una instancia de datos no replicada. Al crear una instancia replicada, el cliente puede incluir al menos un parámetro adicional, tal como un parámetro "Replicado" o similar, con un valor definido como "verdadero" o cualquier otro valor apropiado que indique que la instancia creada debe replicarse. En algunas realizaciones, el valor se ajusta a "falso" de forma predeterminada de tal manera que se creen instancias no replicadas a menos que el cliente especifique otra cosa. En algunas realizaciones, sólo ciertos clientes tienen la capacidad de crear instancias replicadas, tal como un cliente que paga por un cierto nivel de servicio, etc.

En algunas realizaciones, un cliente también puede seleccionar si la réplica secundaria se crea en una zona de datos diferente de la réplica primaria. El cliente en algunas realizaciones también puede permitir seleccionar una o más zonas de datos específicas para las instancias, o una lista ordenada, por ejemplo, mientras que en otras realizaciones, los clientes no pueden seleccionar la zona de datos para al menos la réplica primaria. Si un cliente especifica dos zonas de datos y una de las zonas de datos no está disponible durante un periodo de tiempo prolongado, por ejemplo, los requisitos de durabilidad en algunas realizaciones harán que se genere otra réplica en una tercera zona de datos, etc. Esto podría requerir la gestión y actualización de listas de zonas de datos ordenados para múltiples clientes, lo que puede complicar la experiencia del usuario sin proporcionar ningún beneficio significativo. Además, puede ser más fácil para las aplicaciones distribuir la flota de aplicaciones asociada a través de las zonas de datos, de tal forma que puede haber algunas flotas de aplicaciones ubicadas en la misma zona de datos que la réplica secundaria.

En algunas realizaciones, un cliente puede llamar a una "DescribeDBInstance" o API similar para la instancia de datos replicados, por lo que el RDS puede enumerar información tal como el nombre DNS del punto final de la

réplica primaria y la zona de datos en la que se encuentra actualmente la réplica primaria. Los clientes aún pueden comunicarse con la instancia de RDS utilizando enfoques convencionales que se utilizarán para una sola zona de datos, ya que los clientes pueden recibir el nombre DNS de punto final de un almacén de datos tan pronto como el estado de la instancia de RDS esté "Disponible", por ejemplo, y se conecta a la instancia utilizando el nombre DNS del punto final. En el caso de un fallo de réplica, el RDS puede conmutar por error la base de datos a la réplica secundaria correspondiente, y al nombre de DNS del punto final se le puede dar un alias con respecto a la nueva réplica primaria. El nombre DNS del punto final de la base de datos sigue siendo una constante en muchas realizaciones, sin cambiar durante la vida útil de la instancia replicada.

En algunas realizaciones se puede proporcionar a los clientes la capacidad de convertir una instancia no replicada en una instancia replicada, tal como, llamando a una "ModifyDBInstance" o una API similar con el parámetro Replicado ajustado en "verdadero". Esto puede hacer que la base de datos se convierta en una instancia replicada en un momento apropiado, tal como durante la siguiente ventana de mantenimiento o inmediatamente después de la solicitud, ya que puede depender de los parámetros de llamada de la API, etc.

Varias realizaciones aprovechan un mecanismo de replicación a nivel de bloque, tal como un módulo de núcleo que implementa una solución de almacenamiento replicada sin compartir nada que refleja el contenido de los dispositivos de bloque entre servidores. BLRM funciona en la parte superior de los dispositivos de bloque (es decir, discos duros o volúmenes lógicos). Utiliza una arquitectura de replicación primaria-esclavo en la que la réplica primaria dirige todas las actualizaciones al dispositivo de bloque subyacente. Todas las solicitudes de entrada y salida (I/O) al dispositivo de bloque son interceptadas por el módulo del núcleo BLRM, estando todas las operaciones de escritura replicadas de forma automática y sincrónica. BLRM proporciona la detección inherente de fallos de dispositivos entre pares, e invoca los manejadores de recuperación apropiados cuando un nodo de pares no está disponible. BLRM también puede resincronizar automáticamente un nodo temporalmente no disponible a la última versión de los datos, en segundo plano, sin interferir con el acceso a datos en la réplica primaria. BLRM utiliza identificadores de generación ("GI") para identificar generaciones de datos replicados, por lo que BLRM puede determinar aspectos tales como si los dos nodos son miembros del mismo par de réplicas, la dirección de re-sincronización de fondo (si es necesario), y si es necesaria una re-sincronización parcial o completa. Un controlador BLRM puede iniciar una nueva generación en cualquier momento apropiado, tal como durante la inicialización de un par de réplicas, cuando una réplica en modo en espera desconectada está conmutando a la réplica primaria, o cuando un recurso en la función primaria se está desconectando de la réplica secundaria. Aunque se usa en el presente documento un mecanismo de replicación a nivel de bloques como ejemplo para propósitos de explicación, debe entenderse que puede usarse cualquier otra tecnología o mecanismo a nivel de bloques apropiada dentro del alcance de diversas realizaciones.

Como se analiza, las instancias de datos RDS en diversas realizaciones pueden construirse sobre uno o más sistemas o plataformas. Por ejemplo, las instancias se pueden construir sobre un entorno informático virtual que permite a un cliente utilizar servicios Web u otro enfoque apropiado para iniciar instancias con una variedad de sistemas operativos y gestionar esas instancias. Un ejemplo de un servicio Web que proporciona un entorno de computación virtual es el servicio Elastic Compute Cloud (EC2) ofrecido por Amazon.com, Inc. Las instancias de datos también se pueden construir sobre un mecanismo de almacenamiento a nivel de bloques que puede proporcionar almacenamiento fuera de instancia que persiste independientemente de la vida de un caso. Un mecanismo de almacén de bloques puede proporcionar volúmenes de almacenamiento que pueden conectarse a una instancia y exponerse como un dispositivo dentro de la instancia. Un ejemplo de una plataforma de almacén de bloques se proporciona en la solicitud de patente de Estados Unidos pendiente junto con la presente n.º 12/188.949, presentada el 8 de agosto de 2008, titulada Managing Access of Multiple Executing Programs to a Non-Local Block Data Storage. Se puede construir un volumen lógico (por ejemplo, capa LVM) en los volúmenes de almacenamiento de bloques y un sistema de archivos apropiado, de tal forma que la base de datos del cliente puede ejecutarse en la capa del LVM/sistema de archivos. Para una base de datos replicada en una realización, BLRM puede ejecutarse en la parte superior de la capa LVM. BLRM en tal realización interceptará todas las solicitudes de I/O y enviará esas peticiones al volumen lógico, que a su vez puede dividir las peticiones a través de múltiples volúmenes de almacenamiento de bloques. El uso de un volumen lógico puede proporcionar la capacidad de manejar múltiples volúmenes E de almacenamiento de bloques, así como la capacidad de expandir fácilmente el almacenamiento, etc. La estratificación de BLRM en la parte superior de LVM también puede permitir que las operaciones de escritura se repliquen en las réplicas.

La FIG. 4 ilustra un ejemplo de un mecanismo 400 para implementar un modelo de replicación primaria-secundaria para proporcionar una instancia RDS replicada. En este ejemplo, la réplica primaria 410 y la réplica secundaria 412 están situadas en diferentes zonas de datos (1 y 2) del plano de datos 408, o entorno de base de datos. Cada réplica se construye sobre el mecanismo de almacenamiento de bloques, ilustrado aquí como una

capa BLRM 418, 422 para gestionar I/O en un almacén de bloques 420, 422 para cada réplica. Los componentes del plano de control 406, tal como los que pueden ser similares a los analizados con respecto a la Fig. 2, son capaces de crear la instancia de RDS replicada emitiendo comandos de configuración al administrador de anfitrión local 414, 416, por ejemplo, que puede realizar las operaciones de instalación necesarias. Como se ve en la figura, un mecanismo a nivel de bloque tal como BLRM 418, 422, está posicionado para interceptar todas las peticiones de I/O a nivel de dispositivo de bloques, y escribir información para las solicitudes a los discos locales y los discos remotos 420, 424. En este ejemplo, la base de datos 426 (por ejemplo, SQL) se ejecuta sólo en la réplica primaria 410, y todos los clientes 402 ejecutan sus transacciones de base de datos en la réplica primaria 410 (a través de una red 404 apropiada). La base de datos 426 no se ejecuta en la réplica secundaria 412, y un sistema de archivos también puede no estar montado en la réplica secundaria, ya que la base de datos generalmente no será consciente de las actualizaciones en el dispositivo subyacente.

Cada cliente de base de datos 402 puede descubrir automáticamente la réplica primaria actual utilizando un nombre de punto final DNS de base de datos RDS, que puede dar un alias al nombre de anfitrión de la réplica primaria 410. Usando el DNS para descubrir la réplica primaria actual, la compatibilidad puede mantenerse con clientes de base de datos existentes tal como los clientes nativos MySQL, JDBC, PHP, C# y Haskell, por ejemplo. Aunque el caché de DNS puede hacer que los clientes intenten conectarse a una réplica primaria antigua, un cliente no podrá hablar con la base de datos al conectarse a una réplica secundaria, ya que no se ejecuta ninguna base de datos en la réplica secundaria. El cliente puede entonces saber cómo obtener la información correcta de DNS.

Como se analiza, la replicación de la base de datos puede ser soportada a través de múltiples instancias de datos subyacentes que se ejecutan en la misma o diferentes zonas de datos. Una vez que se valida una operación de escritura utilizando un enfoque síncrono, los datos no se perderán, excepto en el caso extremadamente raro en que todas las réplicas dejan de estar disponibles debido al fallo de múltiples zonas de datos, etc. Tal enfoque puede proporcionar una disponibilidad mayor que una única instancia de base de datos, ya que un fallo de réplica única no causa una interrupción en la base de datos durante un periodo de tiempo prolongado. Por ejemplo, si la réplica primaria de una base de datos está inactiva, el sistema puede realizar una operación de conmutación por error en una réplica secundaria en muchos casos. Además, tal aproximación puede proporcionar una durabilidad mayor que una base de datos no replicada, y puede proteger contra fallos tales como un fallo de una zona de datos o un fallo en el volumen de almacenamiento de un solo bloque, etc.

Como se ha mencionado anteriormente, RDS puede aprovechar un mecanismo a nivel de bloque tal como BLRM para reflejar el contenido de dispositivos de bloque entre servidores. Una arquitectura de replicación primaria-esclavo permite que el primario acepte y escriba todas las actualizaciones al dispositivo de bloque. Todas las solicitudes de I/O al dispositivo de bloque son interceptadas por el módulo del núcleo BLRM, de tal forma que las escrituras pueden replicarse de forma síncrona. BLRM utiliza identificadores de generación ("GI") para identificar generaciones de datos replicados. BLRM utiliza este mecanismo para determinar si dos nodos son de hecho miembros del mismo par de réplicas, en contraposición a dos nodos que se conectaron accidentalmente. Los GI también se pueden usar para determinar la dirección de la resincronización de fondo, si es necesario, y determinar si se necesita una resincronización parcial o completa. En al menos una realización, los GI son universalmente identificadores únicos (UUID) y no son números de secuencia que aumentan monotónicamente. Un controlador BLRM puede iniciar una nueva generación durante la inicialización del par de réplicas, cuando una réplica secundaria desconectada se cambia a la nueva réplica primaria, o cuando un recurso en la función primaria se está desconectando de la réplica secundaria, etc.

En un ejemplo en el que un par de réplicas (por ejemplo, réplica primaria P y réplica secundaria S) se inicializa y se conecta por primera vez, la réplica primaria P puede generar un nuevo GI, tal como GI_1. Si la réplica primaria P se desconecta de S y se mueve a un modo degradado, donde P realiza todas las I/O sin replicación síncrona, P puede generar un nuevo GI, como GI_2. Incluso en el caso en el que P y S estén desconectados debido a una partición de red, sin embargo, S no generará un nuevo GI. En este ejemplo, la réplica primaria P mantiene en sus metadatos los nuevos y los anteriores GI (GI_2 y GI_1, respectivamente). Una razón para almacenar el GI anterior es optimizar la recuperación de la réplica secundaria. Por ejemplo, puede haber una partición de red temporal que hace que S se desconecte momentáneamente. Posteriormente, cuando la partición se arregla y cuando S se vuelve a unir a P, P puede ver que el GI actual de S es el GI anterior para P, de tal forma que P sólo puede enviar los bloques que se cambiaron entre las dos generaciones de datos.

En un ejemplo donde hay un fallo de la réplica primaria, S se puede promocionar a la nueva réplica primaria cuando P se detecta como no disponible. Cuando se emite el comando para promover la réplica secundaria a la nueva réplica primaria, BLRM puede generar un nuevo GI en la nueva réplica primaria (anteriormente S). Por

tanto, cuando P (la réplica primaria original) vuelve a unirse al clúster y se comunica con S, P puede determinar que la generación de datos ha cambiado y P tiene que sincronizar datos de S.

Como se analiza, la réplica primaria P puede aceptar todas las escrituras y lecturas, y el DNS_primary puede dar un alias o cname al nombre DNS de la instancia primaria. La instancia secundaria S puede recibir todas las actualizaciones a través de un protocolo de replicación DRDB (o un replicación a nivel de bloque similar) desde la réplica primaria. No se montan dispositivos ni se inician bases de datos en la réplica secundaria. Cuando se habilita la conmutación por error, otro componente que se puede utilizar es un componente de supervisión M. Un componente de supervisión puede supervisar la integridad de las réplicas primarias y/o secundarias e iniciar acciones de conmutación por error apropiadas cuando se produce un fallo. El componente de supervisión en una realización hace ping periódicamente, o se comunica de otra manera con las réplicas primaria y secundaria. Esta comunicación puede incluir una comunicación de pulsaciones que, por ejemplo, ocurre a intervalos regulares, como un número de segundos especificados por un parámetro T_heartbeat o similar. Siempre que un componente de supervisión hace ping en P y S, el componente de supervisión en una realización envía un comando getStatus( ) de HTTP al administrador de anfitrión que se ejecuta en cada réplica. Cuando P y S reciben la llamada, las réplicas pueden ejecutar una llamada de estado BLRM o similar para determinar el estado actual de cada réplica. Por ejemplo, la réplica primaria P puede ejecutar un comando de herramienta BLRM para determinar el estado, como IN_SYNC, PARALIZADO, DEGRADADO, INACTIVO, etc.

Además de reportar el estado, cada una de las réplicas también puede reportar su GI respectivo al componente de supervisión, que puede almacenar los números de generación en la memoria. Cada vez que se inicia un nuevo arranque de componentes de supervisión, el nuevo componente puede leer la lista de pares de réplicas, así como los puntos finales, a partir de un almacén de datos fuertemente consistente (es decir, la base de datos de supervisión), y almacenar la información en la memoria. Durante cada ping de estado, el componente de supervisión puede determinar si el número es el mismo. Si por alguna razón el número es diferente, el valor GI se puede actualizar en la memoria.

Una réplica primaria o secundaria puede estar en uno de al menos dos estados supervisados. La FIG. 5 ilustra un ejemplo de un diagrama de transición de estado ejemplar 500 para una réplica primaria de acuerdo con una realización. Una réplica puede tener un estado SUPERVISADO cuando la réplica está conectada al componente de supervisión. Una réplica puede estar en un estado NO_SUPERVISADO o similar cuando la réplica no está conectada al componente de supervisión. Una instancia primaria también puede estar en una de una pluralidad de estados de sincronización de datos. Por ejemplo, P puede estar en un estado IN_SYNC cuando tanto P como S están arriba y pueden comunicarse entre sí, donde todas las escrituras están escritas de forma síncrona entre P y S. Viendo el diagrama de estado, en 504, donde la réplica primaria está en un IN_SYNC/supervisado, la réplica primaria puede comunicarse con la réplica secundaria, todas las escrituras tienen éxito, el BLRM está latiendo, y el primario se está supervisando. Si el primario está desconectado del componente de supervisión pero está sincronizado con la réplica secundaria, el estado puede pasar al estado 502. En el estado 502, el primario puede comunicarse con la réplica secundaria y ambas réplicas están conectadas y actualizadas, pero la primaria está desconectada del componente de supervisión y, por lo tanto, no se está supervisando. La réplica secundaria también puede estar en un estado CONECTADO, donde la réplica secundaria está bien y en contacto con la réplica primaria, y puede estar en un estado DESCONECTADO cuando la réplica secundaria está bien pero fuera de contacto con la réplica primaria. Por lo tanto, en los estados 502 y 504, la réplica secundaria estará CONECTADA, pero en los otros estados estará DESCONECTADA.

La réplica primaria puede tener un estado PARALIZADO o similar 508 cuando se supervisa P, pero está desconectada de, o no está en contacto con, S, y no puede continuar con ninguna operación de I/O, ya que todas las escrituras están congeladas. La réplica primaria puede tener un estado DEGRADADO o similar 406 cuando P está desconectado de S y ha cambiado al modo no replicado. Esto permite que P continúe sirviendo lecturas y escrituras cuando S está inactivo o inaccesible de otro modo. P puede alcanzar el modo DEGRADADO desde cualquiera de los estados 502 o 508. P no puede permanecer en modo DEGRADADO durante mucho tiempo en muchas realizaciones, ya que RDS creará típicamente una nueva réplica en modo en espera. Una vez que se ha instanciado un nuevo secundario, está totalmente sincronizado con la réplica primaria, y está siendo supervisado por el componente de supervisión, el estado puede volver al estado 504, donde las réplicas son IN_SYNC y Supervisadas.

La réplica primaria puede estar en un estado SUICIDA o similar 510 cuando P se desconecta de S y también está en, o entra de otra manera, en un estado NO_OBSERVADO. En este caso, el estado de P se puede cambiar a SUICIDA después de un periodo tal como T_failover segundos. Este estado 510 sólo puede alcanzarse desde un estado PARALIZADO 508 en algunas realizaciones, y se produce cuando P está fuera de

contacto con el componente de supervisión. En este estado, la réplica primaria "se mata" por sí misma apagándose, o reiniciando su instancia de datos.

Como parte de una arquitectura de supervisión y conmutación por error para implementar dichos procesos, cada base de datos replicada (es decir, el par de réplicas) es supervisada por un componente de supervisión. En RDS, un único componente de supervisión puede supervisar varios pares de réplicas. Además, el sistema puede utilizar una pluralidad o "flota" de nodos de supervisión. Como se ha analizado, un componente de supervisión puede determinar el estado de una base de datos supervisada haciendo ping continuo en el par de réplicas a intervalos apropiados, tal como cada T_heartbeat segundos. La FIG. 6 ilustra un ejemplo de un diagrama de transición de estado 600 para una base de datos replicada desde el punto de vista de un componente de supervisión respectivo M. Cuando la réplica primaria está en un estado IN_SYNC y el secundario está conectado, M puede ver la base de datos como IN_SYNC o estado similar 604. M también puede ver la base de datos como en un estado 604 cuando el componente de supervisión no puede comunicarse con una de las réplicas debido a una partición de red, por ejemplo, pero la otra réplica indica al componente de supervisión que las réplicas están conectadas y sincronizadas, de forma que no es necesario realizar un evento de conmutación por error.

Si por algún motivo M ya no puede comunicarse con las réplicas primarias y secundarias, el componente de supervisión está dividido o las dos réplicas dejan de estar disponibles al mismo tiempo. En cualquier caso, M puede ver el estado de la base de datos a medida que se mueve a un estado Dividido o similar 602. Esto puede poner tanto la réplica primaria como secundaria en un estado NO_Supervisado. Cuando la partición del monitor se arregla o cuando se asigna un nuevo componente de supervisión a la base de datos, el estado puede volver al estado IN_SYNC 604.

Si M ya no puede comunicarse con la réplica primaria, y la réplica secundaria no puede comunicarse con la réplica primaria de modo que esté en un estado Desconectado, el componente de supervisión puede ver la base de datos en un estado S_ONLY 606. Si, dentro de un periodo de tiempo tal como T_failover segundos, el componente de supervisión es capaz de restablecer comunicaciones con la réplica primaria, el estado puede volver a IN_SYNC 604. Si el monitor no puede comunicarse con la réplica primaria durante al menos T_failover segundos, el componente de supervisión puede decidir promover la réplica secundaria a la nueva primaria. Si la réplica secundaria confirma que el GI actual es el mismo que el último GI conocido de la réplica primaria, y la réplica secundaria confirma la solicitud de promoción, el estado puede hacer la transición a un estado P_ONLY 608 hasta que se instancia un nuevo secundario y se sincroniza completamente con el nuevo primario, momento en el que el estado puede volver de nuevo a IN_SYNC 604.

Sin embargo, si el componente de supervisión decide promover la réplica secundaria a la nueva réplica primaria, pero la solicitud secundaria rechaza la solicitud de promoción, el estado puede hacer la transición a un estado Desastre o similar 610. El secundario podría rechazar la solicitud porque el GI actual para la réplica secundaria es diferente del último GI conocida de la réplica primaria. En otros casos, una respuesta no puede recibir de otra manera a la de la réplica secundaria. Esto podría suceder cuando hay una indisponibilidad masiva, o en el caso altamente improbable de que la información de GI o de membresía se haya corrompido, etc.

En otro caso, en el que el estado es IN_SYNC 604, el componente de supervisión podría perder la capacidad de comunicarse con la réplica secundaria, y la réplica primaria también podría perder la capacidad de comunicarse con la réplica secundaria de manera que la réplica primaria se encuentre en un estado PARALIZADO. En este caso, el componente de supervisión de estado puede solicitar que la réplica primaria se mueva a un estado DEGRADADO, y el estado según se ve por el componente de supervisión puede pasar a un estado P_ONLY o similar 608. Con el componente de supervisión y la réplica primaria incapaces de comunicarse con la réplica secundaria y la réplica primaria en modo DEGRADADO, se puede instanciar una nueva réplica secundaria y sincronizarla completamente con la réplica primaria, por lo que el estado según se ve por M puede hacer la transición de nuevo a IN_SYNC 604.

Como puede verse en los diagramas de transición de estado, un algoritmo de conmutación por error implementado por los componentes de supervisión en al menos una realización puede hacer que un componente de supervisión promueva una réplica secundaria para que sea la nueva réplica primaria para una instancia bajo ciertas circunstancias. Como se comprenderá, este ejemplo representa meramente una trayectoria a través del diagrama de estado de la Fig. 6. La FIG. 7 ilustra un proceso ejemplar 700 para el fallo de una réplica secundaria que se puede usar de acuerdo con una realización. En este ejemplo, las réplicas primaria y secundaria se aprovisionan, se conectan y se sincronizan 702. Se genera un identificador de generación (GI) para cada réplica para identificar la generación actual de datos replicados 704. Se asigna un componente de supervisión a las

réplicas y hace pings periódicamente con las réplicas 706. Un componente de supervisión que se asigna a un par de réplicas puede obtener, o estar dotado de, un "arrendamiento" para ese par, que puede expirar después de un periodo de tiempo. El arrendamiento típicamente se recibirá desde un administrador de anfitrión para la réplica primaria, y un identificador de procesador de eventos y el tiempo de arrendamiento se pueden almacenar en ambas réplicas de tal manera que el esquema de arrendamiento del procesador de eventos pueda sobrevivir a la caída de una réplica primaria. De esta manera, los componentes de supervisión pueden liberarse periódicamente de réplicas, y por lo tanto, pueden moverse a otros pares para propósitos de distribución de carga o partición, o manipularse de otra manera por cualquiera de otras razones diferentes. En o cerca del final de un periodo de arrendamiento, un componente de supervisión puede intentar renovar el arrendamiento, se puede tomar la decisión de no renovar un arrendamiento, etc., como se analiza en otra parte del presente documento. Si el componente de supervisión pierde el contacto con la réplica primaria 708, el componente de supervisión puede hace reintentos durante un periodo de tiempo 710. Si el componente de supervisión vuelve a contactar con el primario en cualquier momento, el proceso de supervisión puede continuar. Si el componente de supervisión está fuera de contacto con la réplica primaria durante un periodo de tiempo tal como T_failover segundos, se determina si la réplica secundaria es capaz de comunicarse con la réplica primaria 712, o si la réplica secundaria está en un estado DESCONECTADO. También se puede determinar si el estado de la réplica primaria en el momento en que se perdió el contacto era conocido como IN_SYNC con la réplica secundaria 714. Las determinaciones pueden realizarse por separado o sustancialmente al mismo tiempo en diversas realizaciones. Si la réplica secundaria no puede comunicarse con la réplica primaria, y las réplicas se sincronizaron (por ejemplo, tenían el mismo valor GI), el componente de supervisión puede emitir un comando para promover la réplica secundaria a la nueva réplica primaria 716. Si el último estado de P no se puede determinar, no se produce una conmutación por error. Un componente de supervisión puede no conocer el estado de P si el proceso o la máquina se reiniciaron, o si se asumió un nuevo componente de supervisión. En ese caso, el estado puede tratarse como DEGRADADO.

Al promover una réplica secundaria para que sea la nueva réplica primaria, un componente de supervisión puede emitir un comando tal como promoteToPrimary (oldGI) en el administrador de anfitrión para la réplica secundaria. En este ejemplo, "oldGI" es el último GI conocido del administrador de anfitrión para la réplica primaria. Tras la recepción de esta solicitud, la réplica secundaria puede probar una última vez para comunicarse con la réplica primaria. Si las réplicas aún no pueden comunicarse, la réplica secundaria verifica que su GI actual es igual que oldGI (de la réplica primaria) 718. La réplica secundaria también puede verificar la información de arrendamiento, por lo que el componente de supervisión que emite la solicitud o envía la solicitud de estado es un componente de supervisión válido para esa réplica, o el actual "titular de arrendamiento" para la réplica. Si es así, la réplica secundaria confirma que puede promocionarse y se convierte en la nueva primaria emitiendo el comando BLRM apropiado 720. La réplica secundaria devuelve la nueva GI al componente de supervisión como una respuesta a la solicitud promoteToPrimary ( ). Posteriormente, el administrador de anfitrión para la nueva réplica primaria (promovida) monta el sistema de archivos e inicia la base de datos (por ejemplo, MySQL) 722. Cuando el componente de supervisión ha promovido con éxito la réplica secundaria, se puede señalar la DNS_primary cname a la nueva réplica primaria 724, como puede realizarse por el componente de supervisión u otro componente del plano de control. Posteriormente, el estado de la instancia se puede marcar como en necesidad de la recuperación secundaria 726.

Sin embargo, si el GI actual de la réplica secundaria no es el mismo que oldGI, puede que no sea seguro promover la réplica secundaria como la nueva réplica primaria. En este caso, el proceso de promoción puede abortarse y generarse una alarma para la intervención del operador (u otra acción correctiva apropiada). Si el operador no puede resolver este problema, se puede realizar una recuperación en un momento dado restaurando la base de datos hasta el último punto bien conocido.

Viendo los diagramas, se pueden determinar varios casos de fallo diferentes. Por ejemplo, en un primer caso de fallo, las réplicas primaria y secundaria se están ejecutando y se están comunicando con un componente de supervisión de funcionamiento. Desde el punto de vista del componente de supervisión, siempre y cuando el componente sea capaz de comunicarse periódicamente con cada instancia, tal como dentro de la mayoría de los segundos del componente de supervisión T, todo se está ejecutando como se esperaba. El estado del primario en este caso será "IN_SYNC/OBSERVADO".

Sin embargo, en el caso de fallo en el que el enlace de red entre el componente de supervisión y la réplica secundaria está dividido, el primario podrá comunicarse con el secundario y el componente de supervisión, pero el componente de supervisión no podrá comunicarse con la réplica secundaria. Desde el punto de vista del primario, todas las escrituras son todavía exitosas de tal manera que el primario está aún en un estado IN_SYNC/OBSERVADO de tal forma que no se inicia la recuperación secundaria. Desde el punto de vista del

componente de supervisión, el componente detecta un fallo secundario, pero el primario sigue sincronizado con el secundario por lo que el componente de supervisión no tiene que realizarse y el funcionamiento y simplemente puede seguir intentando comunicarse con las réplicas.

Si, en cambio, el componente de supervisión no es capaz de comunicarse con el componente principal, como en respuesta a una partición de red, la réplica secundaria podrá comunicarse con la réplica primaria y el componente de supervisión, pero la réplica primaria será inaccesible desde el componente de supervisión. Desde el punto de vista del primario, después de n*T_heartbeat segundos, el primario se moverá a un estado NO_OBSERVADO, ya que la réplica primaria no ha estado en contacto con el componente de supervisión. En algunas realizaciones, el valor de n puede ajustarse a n ≥2. Por lo tanto, el estado del primario puede ser IN_SYNC/NO_OBSERVADO. Desde el punto de vista del componente de supervisión, sólo se puede acceder a la réplica secundaria pero la réplica secundaria sigue aún en contacto con la réplica primaria, de forma que el componente de supervisión no inicie ninguna conmutación por error.

En un caso de fallo ejemplar, la réplica secundaria podría estar inactiva debido a factores tales como un fallo de nodo o partición de red. La FIG. 8 ilustra un ejemplo de un proceso 800 para realizar la recuperación secundaria que se puede usar de acuerdo con al menos una realización. En este ejemplo se asume que las réplicas ya están proporcionadas, comunicándose, y sincronizadas, y las réplicas están siendo supervisadas por un componente de supervisión 802. Si el componente de supervisión pierde el contacto con la réplica secundaria 804, el componente de supervisión puede hace reintentos durante un periodo de tiempo 806. Si el componente de supervisión vuelve a contactar con la réplica secundaria en cualquier momento, el proceso puede continuar. Si el componente de supervisión está fuera de contacto con la réplica secundaria durante un periodo de tiempo, se determina si la réplica primaria puede comunicarse con la réplica secundaria 808. Si la réplica primaria no puede comunicarse con la réplica secundaria, el primario puede entrar en un estado PARALIZADO después de T_sync segundos 810. Después de entrar en el estado PARALIZADO, la réplica primaria puede esperar durante n*T_heartbeat segundos para escuchar al componente de supervisión. Si la réplica primaria escucha al componente de supervisión dentro de esta unidad de tiempo (es decir, el primario está en un estado SUPERVISADO), el primario pasa a un estado DEGRADADO e informa al componente de supervisión en el siguiente intercambio 812. Desde el punto de vista del componente de supervisión, el estado pasa a P_ONLY, donde el componente de supervisión encuentra que la réplica secundaria es inaccesible. Al determinar esto, el componente de supervisión marca el estado de la instancia de base de datos como un estado tal como NECESARIA_RECUPERACIÓN_SECUNDARIA, e inicia un flujo de trabajo de recuperación de réplica secundaria 814, tal como se analiza en otra parte del presente documento.

En otro caso de fallo, todos los anfitriones pueden estar activos y funcionando, pero la réplica primaria puede separarse del componente de supervisión y la réplica secundaria, tal como puede ser debido a una partición de zona de datos o un enlace ascendente de bastidor defectuoso. Por lo tanto, el componente de supervisión es capaz de comunicarse con la réplica secundaria, pero ni el componente de supervisión ni la réplica secundaria es capaz de alcanzar la réplica primaria. Desde el punto de vista de la réplica primaria, después de T_sync unidades de tiempo, la réplica primaria pasa a un estado PARALIZADO. Después de entrar en el estado PARALIZADO, la réplica primaria espera durante n*T_heartbeat segundos para escuchar al componente de supervisión. En este caso, la réplica primaria no escucha al componente de supervisión y se desconecta de la réplica secundaria, de modo que se mueve a un estado SUICIDA y se "mata" reiniciando su instancia cuando vuelve como una réplica secundaria. Desde el punto de vista del componente de supervisión, el componente de supervisión alcanza el estado de S_ONLY, donde encuentra que la réplica primaria no está disponible. El componente de supervisión comprueba con la réplica secundaria en el siguiente intercambio para determinar si la réplica secundaria puede comunicarse con la réplica primaria. En este caso, la réplica secundaria declarará que está en un estado DESCONECTADO. El componente de supervisión espera T_failover segundos y después confirma que la réplica primaria sigue sin estar disponible. Si es así, el componente de supervisión hace que la réplica secundaria sea promovida a ser la nueva réplica primaria, si el estado anterior de la base de datos estaba en IN_SYNC y si el GI actual de la réplica secundaria es el mismo que el último GI conocido de la réplica primaria. El valor de tiempo de T_failover se puede establecer en n*T_heartbeat+T_buffer, donde n es el mismo parámetro descrito previamente en casos anteriores, ajustado en n ≥2. T_buffer es el peor momento esperado para que la réplica primaria se "suicide".

En un caso similar en el que el primario está inactivo y no hay otros problemas, también puede haber una conmutación por error. En este caso, sin embargo, el primario no tiene ningún estado de transición ya que la réplica primaria se ha inactivado y no entrará en un estado SUICIDA u otro estado similar.

En otro caso de fallo, las réplicas primarias y secundarias pueden funcionar y comunicarse como se esperaba,

sin problemas de red, pero el componente de supervisión puede inactivarse o de lo contrario no estar disponible. Desde el punto de vista del primario, todo está todavía en un estado de sincronización de datos IN_SYNC, pero la réplica primaria señala que está en un estado NO_OBSERVADO.

Como se ha analizado, el plano de control incluye un conjunto distribuido de procesadores de eventos, o flotas de procesamiento de eventos, configurados para supervisar las instancias RDS y emitir las acciones de recuperación apropiadas cuando sea necesario. A cada procesador de eventos se le puede asignar una porción de la carga de trabajo de supervisión para una porción de las instancias RDS, tal como empleando un algoritmo de partición basado en hash sencillo en el que el hash se hace basado en un identificador de instancia o un valor de identificación similar. Para supervisar una instancia replicada, un procesador de eventos puede funcionar como componente de supervisión. Un procesador de eventos puede determinar el estado de una instancia de RDS haciendo ping o comunicándose de otro modo con todas las réplicas asociadas con esa instancia. Si una instancia no se replica, el procesador de eventos solo necesita comunicarse con el administrador de anfitrión único para la instancia.

Puede haber consideraciones especiales para dividir la carga de trabajo de supervisión de instancia entre las flotas de procesamiento de eventos cuando hay instancias replicadas. En algunas realizaciones, el sistema de supervisión debería escalar sustancialmente linealmente a medida que aumenta el número de instancias. Este escalamiento puede realizarse en varias instancias añadiendo procesadores de eventos adicionales (por ejemplo, anfitriones). También puede haber restricciones en la colocación del procesador de eventos, ya que puede ser deseable que el procesador de eventos esté situado en una zona de datos diferente de cada una de las réplicas de la base de datos que está siendo supervisada por ese procesador de eventos. Al colocar el procesador de eventos en una zona de datos diferente, el fallo de un centro de datos no da lugar a dos fallos simultáneos (por ejemplo, un fallo del componente de supervisión y al menos una de las réplicas) al mismo tiempo, haciendo que la base de datos alcance potencialmente un estado irrecuperable. También puede ser deseable asegurarse de que cada instancia de base de datos, incluidas todas las réplicas, se supervise continuamente. Esto se puede conseguir en varias realizaciones dividiendo las instancias de base de datos y asignando la propiedad de supervisión de cada partición a uno de los procesadores de eventos. Si un procesador de eventos falla por cualquier motivo, las particiones propias y supervisadas por el procesador de eventos fallido deben redistribuirse de manera uniforme a otros procesadores de eventos disponibles.

Para asegurar la escalabilidad lineal del sistema de supervisión y aún cumplir las restricciones sobre la colocación de los procesadores de eventos, las flotas de procesamiento de eventos, en al menos una realización, se segmentan en diferentes grupos basándose en la zona de datos en la que reside cada flota. Cada grupo puede configurarse de tal manera que los procesadores de eventos dentro de un grupo estén asociados con instancias de RDS cuyas réplicas no estén en la misma zona de datos que el procesador de eventos respectivo.

Como ejemplo, pueden existir cuatro grupos de procesadores de eventos (G1, G2, G3 y G4) que cubren instancias en cuatro zonas de datos respectivas (DZ1, DZ2, DZ3 y DZ4). Para cada par de réplicas, la carga de trabajo de supervisión se puede repartir entre los grupos que no están en las mismas zonas de datos que el par de réplicas. En este ejemplo, la carga de trabajo de supervisión de las instancias de RDS cuyos pares de réplicas están en DZ2 y DZ3 puede dividirse entre los procesadores de eventos en G1 y G4. Para los pares de réplicas en DZ3 y DZ4, la carga de trabajo puede dividirse entre los grupos G1 y G2.

Para todas las bases de datos replicadas ubicadas en una zona de datos determinada, cada procesador de eventos puede calcular la lista de procesadores de eventos que cubren un par de zonas de datos independientemente. Posteriormente, para un par de zonas de datos dado, los identificadores de procesador de eventos que cubren ese par de zonas de datos pueden clasificarse lexográficamente. Los identificadores de base de datos también se pueden clasificar y dividir uniformemente entre los pares de zonas. Por ejemplo, puede haber bases de datos con réplicas en las zonas DZ2 y DZ3. Estas bases de datos pueden ser supervisadas por procesadores de eventos en los grupos G1 y G4 juntos. Para simplificar, los identificadores de base de datos de la base de datos en este par de zonas de datos se pueden establecer como (DB1,…, DB1000), y hay dos procesadores de eventos en el grupo G1 (EP1 y EP2) y dos procesadores de eventos en el grupo G4 (EP3 y EP4), respectivamente. En este ejemplo, cuando EP1 arranca, EP1 puede determinar que hay 1000 bases de datos a supervisar en el par de zonas de datos (DZ2, DZ3) y cuatro procesadores de eventos que los cubren. Mediante la clasificación lexográfica de los identificadores de procesador de eventos, EP1 puede determinar que puede llevar DB1 a DB250, EP2 puede llevar DB251 a DB500, EP3 puede llevar DB501 a DB750 y EP4 puede llevar DB751 a DB1000. EP1 puede repetir las mismas etapas para determinar las bases de datos que EP1 se encarga de supervisar para cada par de réplicas que es idóneo de supervisión.

Para detectar el fallo de un procesador de eventos, cada procesador de eventos puede configurarse para enviar un mensaje de IMPULSOS (por ejemplo, a través de HTTP) a cada otro procesador de eventos periódicamente, tal como cada diez segundos. Los procesadores de eventos también pueden mantener una lista de procesadores de eventos y su estado (por ejemplo, DISPONIBLE o INACTIVO) junto con el último tiempo de registro de cada procesador de eventos. Cuando un primer procesador de eventos no ha escuchado a otro procesador de eventos durante un periodo de tiempo mayor que heartbeat_failure_time, que típicamente es un múltiplo del intervalo de pulsaciones, tal como seis veces el intervalo de pulsación, el primer procesador de eventos puede declarar que el procesador de eventos que no responde está INACTIVO, o en un estado similar, y puede ajustar su carga de trabajo de supervisión. Cuando el anfitrión del procesador de eventos que no responde comienza o se recupera, el procesador de eventos puede iniciarse por sí mismo en un modo de ARRANQUE o similar durante un periodo de tiempo, similar al heartbeat_failure_time, para recibir pulsaciones desde su procesador de eventos de pares, y puede iniciar su agente de pulsaciones. Después de este tiempo, el procesador de eventos puede moverse a un modo OPERATIVO donde determina su porción actual de carga de trabajo de supervisión en base al estado de los procesadores de eventos asignados a su partición. Una razón para dejar los procesadores de eventos en modo ARRANQUE durante un periodo de tiempo es asegurar que el nuevo procesador de eventos que se une al colectivo de procesador de eventos y al procesador de eventos restante tenga tiempo suficiente para converger en el estado actual de los procesadores de eventos activos.

En el caso de un fallo de una zona de datos, es deseable asegurar que las instancias que son supervisadas por los procesadores de eventos en la zona de datos fallida son asumidas por los grupos restantes. En un ejemplo, cuatro grupos de procesadores de eventos (G1, G2, G3 y G4) cubren los procesadores de eventos en cuatro zonas de datos (DZ1, DZ2, DZ3 y DZ4), respectivamente. Si DZ1 muere, la supervisión de instancia por los procesadores de eventos en DZ1 puede ser asumirse automáticamente por los procesadores de eventos en las otras zonas de datos.

Sin embargo, es posible que sólo haya tres zonas de datos en una región, con tres grupos de procesadores de eventos (G1, G2 y G3) supervisando pares de zonas de datos (DZ2, DZ3), (DZ3, DZ1), y (DZ1, DZ2). En el caso de que DZ1 se inactive, G2 y G3 necesitan ser reasignados de tal manera que cada grupo supervise instancias cuya réplica secundaria está en la misma zona de datos que ella misma, con el fin de tolerar el fallo de la zona de datos que contiene la réplica primaria. En diversas realizaciones, un indicador tal como un indicador "anulación-colocación-dz-secundario" puede activarse solamente cuando una zona de datos está fuera en una región de tres DZ. Si este indicador está desactivado, los grupos dividen la carga de trabajo de supervisión con la restricción de que un procesador de eventos no puede residir en la misma zona de datos que los pares de réplicas. Si el indicador está activado, el grupo puede anular la restricción y volver a alinearse para seleccionar las instancias de RDS cuya réplica secundaria esté en la misma zona de datos que ella misma. Este indicador puede persistir en una base de datos de supervisión o almacén de datos similar en el plano de control.

También puede ser deseable asegurarse de que sólo hay un procesador de eventos que supervisa una instancia particular de RDS. En ciertas realizaciones, el algoritmo de conmutación por error requiere que un único componente de supervisión (es decir, el procesador de eventos) supervise un par de réplicas en cualquier momento dado. Esta restricción se puede utilizar porque puede ser indeseable tener dos procesadores de eventos a ambos lados de una partición de red, con un procesador de eventos intentando hacer una conmutación por error en una instancia de RDS y otro suponiendo que el primario siga vivo, lo que lleva a un escenario de "cerebro dividido".

Para asegurarse de que sólo un único procesador de eventos supervisa una instancia de RDS, en algunas realizaciones se puede requerir un procesador de eventos para adquirir explícitamente un arrendamiento de la réplica primaria de una instancia de RDS. En otras realizaciones, el componente de supervisión puede adquirir un arrendamiento de otro componente del entorno de control, que gestiona los arrendamientos e interactúa con los diversos componentes en el entorno de datos. Sólo tras adquirir un arrendamiento de la réplica primaria de una instancia de RDS un procesador de eventos es apto para iniciar la conmutación por error para una instancia de RDS determinada, y sólo para el periodo de arrendamiento tal como arrendamiento T. Un procesador de eventos puede adquirir un arrendamiento de una réplica primaria de instancia de RDS en una realización haciendo ping a una réplica de base de datos (por ejemplo, emitiendo un ping de estado HTTP ( )), por lo que el administrador de anfitrión de la réplica de base de datos puede distribuir un arrendamiento, además de su respuesta habitual. En algunas realizaciones, el arrendamiento se distribuye sólo si la réplica es el primario BLRM, las réplicas primarias y secundarias están sincronizadas, y si todavía hay un arrendamiento válido dado a otro procesador de eventos. Cuando la réplica primaria distribuye el arrendamiento a un procesador de eventos, la réplica primaria puede escribir el tiempo de arrendamiento y el identificador del procesador de eventos en su unidad BLRM. Al escribir en el disco BLRM cuando está sincronizado, la réplica primaria notifica inherentemente

a la réplica secundaria del arrendamiento. Por lo tanto, sólo después de que el tiempo de arrendamiento y el identificador del procesador de eventos se escriban correctamente (es decir, se repliquen en ambas réplicas), la réplica primaria distribuirá un nuevo arrendamiento al procesador de eventos. Además, escribiendo el identificador del procesador de eventos y el tiempo de arrendamiento en ambas réplicas, el esquema de arrendamiento del procesador de eventos es capaz de sobrevivir a la caída de una réplica primaria. La réplica secundaria de una instancia de RDS no entrega ningún arrendamiento en ningún momento en al menos algunas realizaciones. La réplica secundaria puede aceptar una solicitud promoteToPrimary( ) o similar sólo si la solicitud procede del procesador de eventos cuyo identificador es el mismo que el de su unidad BLRM.

Cuando un procesador de eventos se reinicia o un nuevo anfitrión se hace cargo, el procesador de eventos asume que el estado de la instancia de RDS (que no ha supervisado antes) es P_ONLY, un estado en el que la réplica primaria está en modo DEGRADADO. El procesador de eventos hace ping a las réplicas primarias y secundarias para determinar el estado actual de la base de datos y cambia su estado en consecuencia. Como se indicó anteriormente, el procesador de eventos no inicia ninguna conmutación por error si se supone que una réplica primaria está en estado DEGRADADO. Al adoptar un enfoque "pesimista", habrá menos errores cuando un nuevo procesador de eventos asuma el control. Cuando un procesador de eventos se reinicia o un nuevo procesador de eventos toma el relevo, el procesador de eventos hace ping en ambas réplicas asociadas con un anfitrión determinado para determinar qué réplica es la primaria BLRM actual. Una vez recopilada esta información, el procesador de eventos puede comprobar con la API pDNS apropiada para asegurarse de que DNS_primary CNAME señale a la réplica primaria actual. De no ser así, el procesador de eventos puede conmutar por error de inmediato. Este escenario puede ocurrir si un procesador de eventos ha muerto en medio de la conmutación por error. Dado que es posible que la información DNS sea incorrecta debido a la caché de DNS y a otros efectos, la API pDNS puede consultarse sin resolver el nombre DNS, ya que la API pDNS lee la base de datos autorizada. Sin embargo, en el improbable caso de que tanto las réplicas primarias como secundarias crean que son la réplica primaria legítima, el operador o técnico responsable puede ser localizado, etc.

La base de datos de supervisión en el plano de control puede almacenar la lista de las instancias de base de datos activas actuales a supervisar, el tipo de cada instancia (por ejemplo, replicada) y cualquier evento que los procesadores de eventos recopilen para diferentes eventos relacionados con el cliente. A medida que aumenta el número de bases de datos, puede ser necesario en algunas realizaciones escalar más allá de una sola base de datos de supervisión. Para ello, se pueden dividir todas las tablas de la base de datos de supervisión. Para habilitar la división de DB de supervisión, se puede emplear un "mapa de particiones db" junto con los procesadores de eventos. Cuando un procesador de eventos tiene que persistir un evento relacionado con una instancia de base de datos, el procesador de eventos puede consultar el "mapa de partición db" para determinar la base de datos apropiada a la que escribir información para el evento.

La FIG. 9 ilustra un proceso ejemplar 900 para supervisar el estado de los procesadores de eventos en un cubo y gestionar el fallo de uno de los procesadores de eventos de acuerdo con una realización. En este ejemplo, se determina al menos una partición de carga de trabajo para el plano de datos 902. Dependiendo, al menos en parte, del número de almacenes de datos, instancias, administradores de anfitriones y otros componentes a supervisar, la carga de trabajo total se puede dividir en cualquier serie de particiones separadas. Se puede asignar un conjunto de procesadores de eventos a cada partición de carga de trabajo 904, y cada procesador de eventos en el conjunto está asignado a una partición respectiva del trabajo para la partición asignada 906. A intervalos apropiados, cada procesador de eventos envía un mensaje de "pulsación" (por ejemplo, a través de HTTP) a los procesadores de eventos en el mismo conjunto o cubo que cubre la misma partición de carga de trabajo 908. Las pulsaciones se pueden enviar a cualquier intervalo apropiado, tal como cada diez segundos. Una "pulsación" en una realización se refiere a un simple mensaje de multidifusión que se envía a cada procesador de eventos en un cubo para informar a los otros procesadores de eventos del estado del procesador de eventos que envía la pulsación. Los procesadores de eventos pueden mantener una lista de procesadores de eventos y su estado (por ejemplo, "disponible" o "inactivo") junto con el último tiempo de registro de cada procesador de eventos. Si se determina que se recibe una pulsación de cada procesador de eventos en el cubo 910, el proceso puede continuar.

Sin embargo, si se determina que un procesador de eventos en el mismo cubo no ha respondido con una pulsación, entonces se determina si el procesador de eventos no ha podido enviar una pulsación durante un periodo de tiempo igual o mayor que, un tiempo especificado de pulsación (por ejemplo, seis veces el intervalo de la pulsación) 912. Si no se ha alcanzado el tiempo especificado de fallo de la pulsación, el proceso puede continuar. Si se ha alcanzado el tiempo de fallo de pulsación al menos sin una pulsación de un procesador de eventos, cada procesador de eventos activos en el cubo puede declarar que el procesador de eventos que no

responde esta "inactivo", o en un estado similar, y puede reasignar los intervalos de responsabilidad y asumir una parte de la carga de trabajo de supervisión 914. Dado cada procesador de eventos activos en el cubo no recibirá un mensaje de pulsación del procesador de eventos fallido, los procesadores de evento pueden expandir cada uno la carga de trabajo asignada en una cantidad apropiada para recoger el trabajo del procesador de eventos "ausente".

Si hay cuatro procesadores de eventos y se supervisan 60.000 instancias, como se ilustra en el ejemplo 1000 de la FIG. 10, entonces cada procesador de eventos maneja 15.000 instancias (que pueden ordenarse en orden lexográfico u otro orden apropiado por un identificador, etc.). Si falla uno de los procesadores de eventos, los otros tres procesadores de eventos pueden reasignar su respectiva gama de responsabilidad, de modo que cada procesador de eventos maneje ahora 20.000 de las instancias (que siguen siendo ordenadas consecutivamente según el identificador, etc.). Por lo tanto, dado que las instancias se ordenan utilizando un esquema de ordenación, los procesadores de eventos pueden ajustar el intervalo del esquema de ordenación a supervisar, y no tienen que correlacionar ni rastrear de otro modo qué "nuevas" instancias supervisar. Los intervalos que se están supervisando pueden almacenarse en el almacén de datos de supervisión, por ejemplo. Tal enfoque también es beneficioso en situaciones en las que se añaden o eliminan instancias, ya que la carga de trabajo puede distribuirse automáticamente (sustancialmente) uniformemente a través de los procesadores de eventos. La pulsación sólo dentro de un cubo particular también puede ser más eficiente y fácil de mantener que un mecanismo global de pulsación.

La FIG. 11 ilustra un proceso de ejemplo 1100 para reasignar los rangos de trabajo a través de un cubo cuando se añade un procesador de eventos al cubo, tal como puede ser el resultado de añadir capacidad de procesamiento adicional o un resultado de recuperación de un procesador de eventos fallido y de nuevo poder manejar una porción de la carga de trabajo. Un procesador de eventos puede convertirse en activo 1102, tal como un anfitrión de procesador de eventos reiniciándose o recuperándose, o simplemente activándose o añadiendo el anfitrión a un cubo. El procesador de eventos también se puede añadir al cubo 1104, aunque en casos de recuperación, el procesador de eventos podría ya estar asignado a dicho cubo. Cuando el procesador de eventos activo se añade al cubo, el administrador de eventos puede entrar en un modo tal como un modo de "arranque" durante un periodo de tiempo (por ejemplo, el tiempo de fallo de la pulsación) para recibir pulsaciones de los procesadores de eventos en pares en el cubo 1106, para obtener información sobre los otros procesadores de eventos activos en el cubo y determinar un tiempo para enviar pulsaciones, por ejemplo. El procesador de eventos puede acoplarse a un agente de pulsaciones para iniciar también el envío de pulsaciones a los otros procesadores de eventos en el cubo 1108. Después de este tiempo, el anfitrión puede moverse a un modo "operativo", donde cada procesador de eventos puede reasignar el intervalo de trabajo y determina su porción actual de carga de trabajo de supervisión en base al estado de los procesadores de eventos asignados a su partición 1110. Una razón para dejar los procesadores de eventos en modo "arranque" durante un periodo de tiempo es asegurar que el nuevo procesador de eventos que se une (o se une de nuevo) al colectivo de procesador de eventos y los procesadores de eventos restantes tengan tiempo suficiente para converger en el estado actual de los procesadores de eventos activos.

Un enfoque de acuerdo con una realización también sobre-particiona los procesadores de eventos, tal como, por ejemplo, ejecutando cada procesador de eventos a un 50-60 % de capacidad. Tal enfoque permite que al menos uno o dos procesadores de eventos fallen en cada cubo sin tener un impacto significativamente negativo en el rendimiento. Un procesador de eventos fallido eventualmente volverá a estar disponible de nuevo, tal como cuando el anfitrión respectivo se reinicie. Ese procesador de eventos puede entonces comenzar a intercambiar las pulsaciones nuevamente, por lo que los otros procesadores de eventos en el cubo pueden detectar automáticamente la presencia del procesador de eventos. El trabajo asignado se puede redistribuir automáticamente como se ha analizado anteriormente, de modo que el trabajo se distribuya relativamente uniformemente a través del conjunto más grande de procesadores de eventos disponibles en el cubo.

Además de los casos de fallo analizados anteriormente, pueden existir otros modos de fallo que pueden ser tratados de acuerdo con las diversas realizaciones. Por ejemplo, una instancia de réplica primaria podría reiniciarse, de manera que cuando el administrador de anfitrión para el primario vuelva a estar en línea, primero encontrará que el estado BLRM ha cambiado de "primario/secundario" a "secundario/secundario", ya que la réplica primaria vuelve a estar en línea como una réplica secundaria si el componente de supervisión no ha fallado ya en la réplica secundaria. A continuación, puede corresponder al procesador de eventos (por ejemplo, el componente de supervisión) determinar quién debe ser el primario entre las dos réplicas y realizar la llamada promoteToPrimary( ) apropiada. Si se reinicia una instancia de réplica secundaria, el componente de supervisión apreciará que la secundaria está fuera y puede marcar la instancia para la recuperación. Sin embargo, mientras tanto, si la réplica secundaria vuelve a estar en línea (después de reiniciar), el flujo de trabajo de recuperación

secundario puede notar esto y solicitar que el administrador de anfitrión de la réplica secundaria intente volver a conectarse. Esto puede evitar el coste de crear una nueva réplica secundaria para un escenario de reinicio de instancia simple. Si se reinicia una instancia no replicada, el administrador de anfitrión puede convertir automáticamente su estado de una réplica secundaria a una réplica primaria sin necesidad del componente de supervisión para promover la instancia. Esto puede reducir el tiempo de recuperación, por ejemplo, el reinicio para una instancia no replicada.

Cuando una réplica primaria falla y no vuelve a estar en línea, el componente de supervisión puede detectar el fallo principal y promover que la réplica secundaria sea la nueva primaria. Posteriormente, el componente de supervisión puede marcar el estado de la instancia de RDS en el almacén de datos de Admin para que se encuentre en un estado tal como "PENDIENTE/DEGRADADO_NECESARIA_RECUPERACIÓN_SECUNDARIA". Este estado puede hacer que un rastreador de recuperación inicie un flujo de trabajo de recuperación adecuado. El flujo de trabajo de recuperación puede intentar determinar si ambas réplicas están vivas. Si la réplica primaria antigua ha vuelto a estar en línea como una réplica secundaria, tal como, por ejemplo, cuando el reinicio se tomó un tiempo suficiente tal que el componente de supervisión marcó la réplica como inactiva, el flujo de trabajo puede conectar la réplica primaria antigua a la nueva primaria y marcar la recuperación realizada, tal como con un estado de base de datos de OK, una vez que las réplicas están totalmente sincronizadas. Sin embargo, si la primaria antigua no ha vuelto en absoluto, el flujo de trabajo puede terminar la antigua instancia y rotar una réplica secundaria utilizando las mismas etapas descritas con respecto a la creación de una instancia replicada. Si la réplica secundaria falla, el componente de supervisión puede detectar el fallo y marcar el estado de la instancia en el almacén de datos de Admin para que se encuentre en un estado en el que se inicie el flujo de trabajo de recuperación, tal como, por ejemplo, usando un estado "PENDIENTE/DEGRADADO_NECESARIA_RECUPERACIÓN_SECUNDARIA" o similar. Cuando la base de datos se cae por alguna razón, el administrador de anfitrión de la réplica primaria puede actuar como el proceso de niñera y reiniciar la base de datos automáticamente.

Como se ha analizado, cada partición de la carga de trabajo de supervisión puede ser cubierta por un conjunto de procesadores de eventos. Cubrir una sola partición de la carga de trabajo con un conjunto de procesadores de eventos permite redistribuir la carga de supervisión a través de los procesadores de eventos restantes en el caso de que uno de los procesadores de eventos falle o experimente cualquiera de una diversidad de otros problemas diferentes. En una realización, cada grupo de procesadores de eventos está contenido en un cubo u otra partición de este tipo. Cada procesador de eventos en un cubo es responsable de manejar un rango de instancias en un único plano de datos, o agrupación de instancias en ese plano. Se puede utilizar un proceso de detección de fallos para asegurar que si ocurre un fallo, los otros procesadores de eventos de ese contenedor asumen la responsabilidad de las instancias manejadas por el procesador de eventos fallido. El almacén de datos de supervisión en al menos una realización contiene la lista de instancias de datos activos actuales a supervisar por el conjunto de procesadores de eventos en un cubo, así como la información que los procesadores de eventos recogen para diversos eventos relacionados con el cliente. A medida que aumenta el número de instancias supervisadas, puede ser necesario escalar más allá de un único almacén de datos de supervisión. Por lo tanto, cada tabla en el almacén de datos de supervisión se puede dividir, incluyendo la db_poll_list.

En una realización, los procesadores de eventos se despliegan con una tabla de partición del siguiente formato ejemplar:

ID de partición	*Rango de Hash*
P0	0-10000
P1	10000-20000

Esta configuración de partición se puede implementar como un archivo de configuración en los anfitriones del procesador de eventos.

Si una partición de carga de trabajo determinada genera un número significativo de eventos que deja al conjunto responsable de procesadores de eventos en un modo de recuperación constante (es decir, no puede finalizar los controles de estado asignados dentro de un periodo de tiempo determinado), se pueden añadir procesadores de eventos adicionales al conjunto responsable de esa partición de carga de trabajo sin tener que volver a dividir el almacén de datos. Utilizando tal técnica, la escalabilidad de rendimiento se puede diferenciar de los problemas de escalabilidad de datos. Por ejemplo, una única partición que genera tantos eventos que los procesadores de eventos no pueden alcanzar puede distinguirse de una situación en la que la única partición genera tantos eventos que un único almacén de datos no proporciona suficiente espacio de almacenamiento.

La pertenencia de los procesadores de eventos y las particiones a las que están asignados los procesadores de eventos se pueden almacenar en una ubicación tal como un archivo de configuración de pertenencia a un procesador de eventos. La información de configuración de pertenencia se puede implementar en los procesadores de eventos de un grupo (tal como en la misma partición o cubo), y puede tener el siguiente formato ejemplar:

<Identificador EP> <Nombre Anfitrión EP> <puerto_puntofinal> <Id de partición>

Cuando una sola partición está cubierta por múltiples procesadores de eventos, cada procesador de eventos divide las gamas de cubo ordenando los identificadores del procesador de eventos, por ejemplo, utilizando una rutina de ordenación lexográfica o basada en hash, y dividiendo los rangos de cubo uniformemente. Cada procesador de eventos puede determinar independientemente el rango apropiado a supervisar.

En dicho sistema, también puede ser importante asegurarse de que la lista o el conjunto de almacenes de datos y/o instancias a supervisar se rellenan y actualizan automáticamente con el tiempo. Un enfoque sería crear una tabla de listas de base de datos, por ejemplo, que es una réplica de instantánea de las instancias que pueden propagarse según sea necesario. Tal enfoque, sin embargo, puede ser difícil de mantener, así como asegurar que cada componente apropiado tenga la copia más reciente. Otro enfoque sería que los procesadores de eventos consulten los componentes del plano de datos y, a continuación, almacenen la información localmente en el plano de control. Este enfoque puede crear una gran cantidad de tráfico de mensajería, y puede ser difícil de mantener y actualizar. Un enfoque de acuerdo con una realización en su lugar permite a cada procesador de eventos exponer una interfaz tal como un "setStatus" o API similar. Como parte de un flujo de trabajo de "crear" o "eliminar", por ejemplo, se puede añadir una tarea al final del flujo de trabajo que ordena al administrador de anfitrión adecuado que llame al procesador de eventos que es, o era, el encargado de gestionar la instancia. El administrador de anfitrión puede llamar a la API "setStatus" del procesador de eventos para establecer un estado del anfitrión, cada vez que se produzca un cambio de estado como resultado de un flujo de trabajo (u otra acción similar). Cada vez que un procesador de eventos recibe una llamada a través de la API "setStatus", la información se puede poner en un almacén de datos local para añadir el nuevo anfitrión a su conjunto de particiones, eliminar el anfitrión, etc. La información para el anfitrión también puede escribirse en el almacén de datos de supervisión u otra ubicación persistente apropiada.

En una realización, una lista autorizada de instancias de datos activos actuales reside en el almacén de datos de Admin. Una lista activa de instancias de datos a supervisar se encuentra en el almacén de datos de supervisión en una tabla tal como una tabla "db_poll_list". Para añadir, eliminar o actualizar el estado de una instancia en el almacén de datos de supervisión, los procesadores de eventos exponen una API "updateHost" que acepta parámetros tales como un identificador de almacén de datos, parámetros relacionados con una instancia de datos (por ejemplo, un identificador de instancia y una dirección DNS) y un estado de instancia (por ejemplo, "añadir", "eliminar" o "actualizar"). Cuando un procesador de eventos recibe esta llamada, el procesador de eventos hace los cambios apropiados (por ejemplo, añadiendo, eliminando o actualizando una entrada) a la tabla db_poll_list. Por ejemplo, si un cliente envía una solicitud para crear un almacén de datos con un identificador de almacén de datos "id1", el flujo de trabajo para crear el almacén de datos, tras el aprovisionamiento de los recursos necesarios y la configuración del almacén de datos, marcará el estado de id1 como "disponible" en el almacén de datos de Admin. Como última etapa en la creación de la tarea de flujo de trabajo de base de datos, se puede invocar la API updateHost en uno de los procesadores de eventos, tal como llegando a través de una IP virtual interna, para añadir el almacén de datos (y sus instancias) al flujo de trabajo de supervisión. Haciendo que la actualización del estado de supervisión sea la etapa final (o al menos casi final) del flujo de trabajo de aprovisionamiento, la disponibilidad de la creación, eliminación o modificación de un almacén de datos RDS se desacopla de la disponibilidad del almacén de datos de supervisión.

Una vez que el administrador de anfitrión establece el estado de una instancia activa a supervisar, el procesador de eventos responsable puede hacer ping periódicamente al administrador de anfitrión para la instancia como se analiza en otra parte del presente documento. Si una instancia no está disponible, tal como puede deberse a que una máquina anfitriona se ha caído o reiniciado, el procesador de eventos no obtendrá una respuesta para la instancia y escribirá información para el problema potencial en el almacén de datos de Admin. Un rastreador detectará la información y hará que se genere y ejecute un flujo de trabajo de recuperación apropiado. En una realización, un flujo de trabajo de recuperación examina primero el historial de métricas para un almacén de datos o una instancia de datos, tal como información que detalla un historial de errores de I/O para una instancia. El flujo de trabajo intenta después determinar automáticamente si la instancia está inactiva, por ejemplo, dónde hay errores de conexión o si no hay problemas de conexión, sino un número creciente de errores de I/O, lo que

indica un problema potencial con un volumen particular que soporta la instancia. Las tareas del flujo de trabajo pueden intentar determinar y/o aislar automáticamente el problema, donde hay una serie de problemas diferentes que pueden ocurrir para una serie de componentes diferentes. Tal determinación, así como la recuperación de tales problemas, no es una cuestión trivial.

Sin embargo, puede haber situaciones en las que no sea deseable recuperarse automáticamente de un fallo. Por ejemplo, es posible que falle un centro de datos entero, donde miles de almacenes de datos no están disponibles. Puede ser indeseable intentar recuperar todos estos almacenes de datos sustancialmente al mismo tiempo. En una realización, el rastreador (u otro componente del plano de control) puede configurarse con un número máximo de errores o ejecutar simultáneamente flujos de trabajo de un tipo particular. Si un número de flujos de trabajo excede un número o umbral especificado, por ejemplo, un mensaje u otra notificación de este tipo puede enviarse o generarse de otro modo para un operador o DBA, por lo que un usuario experimentado puede determinar el mejor enfoque para resolver la situación. En una realización, el rastreador ejecutará como máximo un número determinado de flujos de trabajo del mismo tipo en cualquier momento dado, tal como diez flujos de trabajo de un tipo dado, pero no generará una alarma hasta que un segundo número, tal como veinticinco, o flujos de trabajo del mismo tipo, se soliciten. Un sistema de acuerdo con una realización proporciona un panel de servicio operacional en el que un DBA u otro operador autorizado puede evaluar el estado de uno o más procesos de supervisión y puede ejecutar manualmente acciones de recuperación. Utilizando dicha interfaz, un DBA puede seleccionar opciones que permiten iniciar los flujos de trabajo, como se analiza en el presente documento, para realizar acciones de recuperación específicas. La interfaz puede utilizarse con el plano de control para trabajar con múltiples motores y sistemas de bases de datos dispares, aunque el plano de control no esté en la ruta de datos del plano de datos. El plano de control puede supervisar mensajes de error y registros, por ejemplo, para cada uno de los motores. Este enfoque también puede permitir que cada almacén de datos sea supervisado como un conjunto, supervisando al mismo tiempo cualquier réplica del almacén de datos. Se puede realizar una recuperación diferente en función del estado de las réplicas, etc.

Debe reconocerse que puede haber una diversidad de tipos de fallos que pueden dar como resultado la indisponibilidad o no fiabilidad de un almacén de datos o una instancia de datos. Por ejemplo, un dispositivo anfitrión puede fallar o reiniciarse, o puede haber un problema con la aplicación de administrador de anfitrión que gestiona la instancia. También puede haber un problema con el almacén de datos, como un volcado de núcleo o excepción de violación de segmentación (SegV). También puede haber problemas con las operaciones de I/O o rutas de comunicación, o un fallo de la instancia que aloja el almacén de datos. También puede haber diversos tipos de fallos diferentes, tal como un fallo de un volumen lógico, una interrupción de la red, o un fallo de la zona de datos. Pueden utilizarse diferentes flujos de trabajo para intentar determinar y recuperarse de los diferentes tipos de fallos. En un ejemplo, el administrador de anfitrión en una realización es la pasarela a una instancia de datos respectiva, y el fallo de este administrador de anfitrión esencialmente no permite ningún control sobre esa instancia. Para solucionar fallos como un proceso de Tomcat que queda sin memoria, un componente de supervisión del plano de control puede asegurar que Tomcat se reinicie si es necesario. El sistema de supervisión puede coordinar los reinicios para evitar errores o detección de errores innecesarios.

Además, como se analiza, no basta simplemente con detectar y recuperarse de un fallo, ya que hay que considerar otros factores, tal como el tamaño o la escala del fallo. Por ejemplo, la acción de recuperación para el fallo de una sola instancia de nube que aloja un almacén de datos puede ser sustancialmente diferente de una acción de recuperación que se ocupa del fallo de una zona de datos completa. Para problemas más grandes, es posible que sea necesario correlacionar y analizar los fallos múltiples de tal manera que las acciones de recuperación no agraven los problemas existentes intentando recuperar simultáneamente las distintas instancias individualmente. En algunos casos, puede ser deseable realizar una recuperación escalonada, donde no sólo se limita el número de procesos concurrentes, sino que se puede controlar el ordenamiento de los procesos de tal manera que no se pierden datos y no se toman medidas de recuperación que más tarde se necesitarán corregir debido a las acciones de recuperación posteriores. También puede ser deseable en algunos casos localizar el proceso de recuperación tanto como sea posible. Puede ser beneficioso en al menos algunas realizaciones tratar un fallo localmente de una manera segura, cuando sea posible. Por ejemplo, las acciones de recuperación local para fallos simples tal como un fallo de un administrador de anfitrión o un proceso de datos, se pueden preferir a una acción realizada por una pila de Admin del sistema RDS general.

También puede haber varias razones para que falle una instancia de datos, un almacén de datos o un proceso de I/O, cada uno de los cuales puede requerir una acción de recuperación diferente. Por ejemplo, un error de almacén de datos puede hacer que el almacén de datos falle, o al menos generar un número significativo de errores de lectura/escritura. Un almacén o instancia de datos también puede fallar debido a sobrecargas, bloques defectuosos u otras situaciones similares. También puede haber errores inducidos por el usuario, tal como una

consulta incorrecta que provoca el bloqueo del almacén de datos. En otros casos, un volumen de registro de almacén de datos puede estar lleno o dañado. Para abordar estos y otros tipos de fallos, los procesos de datos pueden ser supervisados constantemente por el administrador del anfitrión. Como se ha comentado, cada administrador de anfitrión puede tener un componente de supervisión de estado que compruebe el estado del almacén o instancia de datos, tal como, ejecutando un comando *get status* (por ejemplo, para MySQL puede adoptar la forma de estado /bin/mysql_admin). El componente de supervisión de estado puede comprobar periódicamente el estado y, si una instancia no está disponible, la instancia puede reiniciarse o tratarse de otra forma. Si una instancia no está disponible repetidamente o experimenta otros errores, el componente de supervisión de estado puede dejar de intentar corregir el error y hacer que la información se escriba en un almacén de datos de supervisión o Admin en el plano de control.

Para detectar errores del almacén de datos y bloqueos de I/O, el registro de errores del almacén de datos y/o el registro del núcleo se pueden supervisar en algunas realizaciones. Cada administrador de anfitrión puede ejecutar otro módulo que explora continuamente ciertos tipos de error en estos dos (u otros) registros de errores, y genera las métricas relacionadas. Para cada tipo de error, se puede establecer un umbral predefinido, más allá del cual los errores serán enviados a un operador para su análisis y posible recuperación.

Un mecanismo de detección de fallos de acuerdo con una realización tiene una serie de restricciones aplicadas. Por ejemplo, puede configurarse que los componentes de supervisión escalen linealmente, de tal manera que cuando el número de instancias de datos excede el número de anfitriones, un cubo de procesadores de eventos se establece para sondear, por ejemplo, se pueden añadir componentes de supervisión adicionales según se desee. Además, puede establecerse que todas las instancias de datos deben supervisarse constantemente, tal como, por ejemplo, dividiendo las instancias de datos y asignando la propiedad de supervisión de cada partición a uno de los procesadores de eventos. Como se analiza, si un procesador de eventos falla por cualquier motivo, las particiones propias y supervisadas por el procesador de eventos fallido pueden redistribuirse de manera uniforme a otros procesadores de eventos disponibles, tal como procesadores en el mismo cubo. Además, una lista de instancias de base de datos puede mantenerse actualizada añadiendo tareas a flujos de trabajo como clientes de RDS, crear y eliminar almacenes de datos y/o instancias.

Partición de almacenes de datos

Como se conoce bien en sistemas distribuidos altamente escalables, la partición dentro de un almacén de datos sólo se escala con respecto a los límites del sistema físico en el que reside el sistema de almacén de datos. Debido a esta limitación, puede ser deseable estructurar el sistema de tal manera que el sistema pueda escalar tanto dentro de un único sistema de almacenamiento de datos, como a través de muchos sistemas de almacenamiento de datos. La división horizontal de datos a través de diferentes sistemas de almacenamiento de datos puede contribuir a un sistema altamente escalable que puede manejar demandas significativas en el almacenamiento de eventos.

Un sistema de acuerdo con una realización utiliza un identificador de cliente como la clave de partición para dividir las tablas de datos, incluyendo la lista de instancias de base de datos (db_poll_list), los eventos relacionados (tabla db_events), y la tabla de eventos del grupo de seguridad. Puede ser ventajoso utilizar un identificador de cliente sobre un identificador de almacén de datos, ya que algunos eventos no están restringidos a un único almacén de datos e incluso pueden no afectar a un almacén de datos particular. Por ejemplo, un cambio en un grupo de seguridad no se aplica directamente a ningún almacén de datos, pero puede ser necesario almacenarlo como un evento visible del cliente (es decir, recuperable mediante una API de DescribeEvents). Además, los eventos de un único cliente pueden no crecer más allá del espacio de almacenamiento de un único almacén de datos, ya que en algunas realizaciones, los datos de eventos sólo se conservan durante un periodo de tiempo limitado, tal como durante catorce días.

Hay varias maneras de manejar la división de conjuntos de datos a través de particiones horizontales de almacén de datos, por ejemplo, mediante el uso de la partición de cubos La división de cubos proporciona una capa de abstracción entre los datos que se dividen y las particiones donde se almacenan los datos. Esta capa de abstracción permite una gestión operativa más fácil de las particiones, como la adición de nuevas particiones con una migración de datos a lo largo del tiempo, al tiempo que permite que la aplicación utilice un mecanismo de hashing para determinar la colocación de datos divididos. La implementación del sistema de partición de cubos como se describe en la presente invención comprende componentes que son específicos de ciertas realizaciones, pero el concepto general es aplicable a muchos casos de uso diferentes como debería ser evidente.

Para implementar la partición de cubo, se puede determinar un número fijo de cubos que deben estar disponibles para una aplicación. El número de cubos puede permanecer fijo durante la vida útil de la aplicación, de tal manera que la elección de un número suficientemente grande puede ser importante en ciertas realizaciones. El número de cubos puede reflejar la capacidad de distribuir uniformemente la carga en todos los cubos, que pueden asignarse individualmente a un número menor de particiones físicas. Si hay demasiadas instancias individuales asignadas al mismo cubo, puede resultar problemático entonces almacenar de forma eficiente varios cubos en una sola partición. El número fijo de cubos puede actuar como una capa intermedia entre los datos que se van a dividir y las propias particiones. Una primera etapa en la estratificación es averiguar cómo son de diferentes las piezas de datos que se correlacionan con los distintos cubos. Como se ha mencionado anteriormente, la clave de partición para los datos puede ser el identificador de cliente. Se puede utilizar un algoritmo de hashing eficiente y consistente para proporcionar un valor que se puede asignar directamente a un cubo individual. Siempre que un identificador de cliente autentique un valor asignado a un cubo, ese identificador puede residir en ese cubo durante la vida útil de los datos.

En este ejemplo, los cubos se asignan a particiones de carga de trabajo individuales. Siempre puede haber más cubos que particiones, por lo que se puede utilizar una correlación para asignar muchos cubos diferentes a las particiones individuales. Para hacer la configuración de asignación concisa, pueden utilizarse intervalos de los números de cubo para asignar los cubos a las particiones individuales. A continuación se muestra una tabla de ejemplo que muestra cómo puede funcionar la asignación de partición:

Partición 1 = {1-25000}

Partición 2 = {25001-50000}

En este ejemplo, los números de cubo 1 a 25.000 se asignan a la "Partición 1", mientras que los números de cubo 25.001 a 50.000 se asignan a la "Partición 2". Cada vez que se necesita agregar datos al sistema y el hash del identificador de cliente correlaciona la instancia de flujo de trabajo con el cubo 100, por ejemplo, cualquier dato relacionado con ese cliente (incluidos almacenes de datos y grupos de seguridad) puede insertarse en tablas que se encuentran físicamente en la "Partición 1". Dicho enfoque también se puede utilizar para leer cualquier información relativa a una base de datos o grupos de seguridad de un cliente, donde se leerá una solicitud de los eventos para un cliente determinado cuyo identificador autentica el cubo 100 que será leído de la "Partición 1".

El ejemplo anterior se ocupa de un caso relativamente sencillo, con la asignación inicial de los cubos a las particiones sin cambios. A veces, sin embargo, una nueva partición tendrá que añadirse al sistema para aliviar la carga de las otras particiones. Utilizando este ejemplo anterior, se puede agregar una nueva partición "Partición 3" para extraer la carga de las otras dos particiones:

Partición 1 = {1-16666}

Partición 2 = {33333-50000}

Partición 3 = {16667-33333}

Como puede parecer, 8334 cubos (números 16667 a 25000) se han tomado de la "Partición 1" y se han reasignado a la "Partición 3". Además, 8333 cubos adicionales (números 25001 a 33333) se han tomado de la "Partición 2" y se han reasignado a la "Partición 3". Esta reasignación podría haberse basado en los cubos que estaban más ocupados o más llenos, pero en este ejemplo había una redistribución relativamente uniforme de los cubos por todas las particiones.

A medida que cambia la asignación de cubos, los datos que residen en la partición física pueden verse afectados. En un ejemplo anterior, se utilizó el cubo 100 para almacenar la información para un cliente cuyo identificador se había autenticado en 100. En este escenario de reparto, los datos no se verían afectados puesto que el cubo 100 permanecía en la "Partición 1". Sin embargo, puede haber datos en el cubo 11000 y cualquier información escrita antes del reparto reside en la "Partición 1", pero cualquier dato escrito después del reparto existirá en la "Partición 3". Para resolver este problema con datos anteriores existentes en una partición y datos actuales existentes en otra partición, el sistema puede permitir que se asigne más de una partición a un cubo. Un cubo dado puede tener al menos dos particiones, una partición actual y una partición anterior. En el presente ejemplo, el reparto dará lugar a los cubos 10001 a 15000 que tienen dos particiones asignadas, con la "Partición 3" como la partición actual y la "Partición 1" como partición anterior. Como se menciona, cualquier nuevo dato

para el cubo 11000 estará en la partición actual, mientras que cualquier dato escrito antes del reparto estará en la partición anterior. Cuando una consulta de eventos o cualquier información se correlaciona con el cubo 11000, puede ser importante comprobar la partición actual para esos datos, así como comprobar la partición anterior, ya que el registro también podría existir allí. Tal soporte para las búsquedas de particiones múltiples en un cubo puede incurrir en el coste potencial de errores para aquellas instancias que terminan en la partición anterior para un cubo dado. Sin embargo, dado que todos los eventos recién creados se están escribiendo en la partición actual, el coste de un fallo sólo se incurrirá para las instancias de flujo de trabajo que se ejecutan cuando se realiza el reparto o para flujos de trabajo cerrados. El favorecer los eventos recién creados puede mejorar el rendimiento al tiempo que permite la flexibilidad para hacer un reparto eficiente.

Como se ha analizado anteriormente, las diversas realizaciones pueden implementarse en una amplia variedad de entornos operativos, que en algunos casos pueden incluir uno o más ordenadores de usuario, dispositivos informáticos, o dispositivos de procesamiento que se pueden usar para operar cualquiera de varias aplicaciones. Los dispositivos de usuario o cliente pueden incluir cualquiera de varios ordenadores personales de uso general, tal como ordenadores de escritorio o portátiles con un sistema operativo estándar, así como dispositivos celulares, inalámbricos y portátiles que ejecutan software móvil y son capaces de soportar una serie de redes y protocolos de mensajería. Dicho sistema también puede incluir una serie de estaciones de trabajo que ejecutan cualquiera de una diversidad de sistemas operativos comercialmente disponibles y otras aplicaciones conocidas con propósitos tales como desarrollo y administración de bases de datos. Estos dispositivos también pueden incluir otros dispositivos electrónicos, tales como terminales ficticios, clientes ligeros, sistemas de juego y otros dispositivos capaces de comunicarse a través de una red.

También pueden implementarse varios aspectos como parte de al menos un servicio o servicio Web, tal como puede ser parte de una arquitectura orientada a servicios. Los servicios como los servicios Web pueden comunicarse utilizando cualquier tipo de mensajería apropiada, tal como el uso de mensajes en formato de lenguaje de marcado extensible (XML) e intercambiados utilizando un protocolo apropiado tal como SOAP (derivado de "protocolo simple de acceso a objetos"). Los procesos proporcionados o ejecutados por dichos servicios pueden escribirse en cualquier idioma apropiado, como el lenguaje de descripción de servicios web (WSDL). El uso de un lenguaje tal como WSDL permite la funcionalidad, tal como la generación automatizada de código del lado del cliente en varios marcos SOAP.

La mayoría de las realizaciones utilizan al menos una red que será conocida para los expertos en la técnica para soportar comunicaciones usando cualquiera de una diversidad de protocolos comercialmente disponibles, tales como TCP/IP, OSI, FTP, UPnP, NFS, CIFS y AppleTalk. La red puede ser, por ejemplo, una red de área local, una red de área amplia, una red privada virtual, Internet, una intranet, una extranet, una red telefónica pública conmutada, una red de infrarrojos, una red inalámbrica y cualquier combinación de las mismas.

En las realizaciones que utilizan un servidor Web, el servidor Web puede ejecutar cualquiera de una diversidad de aplicaciones de servidor o de nivel intermedio, incluidos servidores HTTP, servidores FTP, servidores CGI, servidores de datos, servidores Java y servidores de aplicaciones empresariales. Los servidores también pueden ser capaces de ejecutar programas o secuencias de comandos en solicitudes de respuesta de dispositivos de usuario, tales como ejecutando de una o más aplicaciones Web que pueden implementarse como una o más secuencias de comandos o programas escritos en cualquier lenguaje de programación, tales como Java®, C, C# o C++, o cualquier lenguaje de secuencias de comandos, tales como Perl, Python o TCL, así como combinaciones de los mismos. El servidor o servidores también pueden incluir servidores de base de datos, incluyendo, sin limitación, los comercialmente disponibles de Oracle®, Microsoft®, Sybase® e IBM®.

El entorno puede incluir una diversidad de almacenes de datos y otros medios de almacenamiento y memoria como se ha explicado anteriormente. Estos pueden residir en una diversidad de ubicaciones, tal como en un medio de almacenamiento local de (y/o residir en) uno o más de los ordenadores o remotos de cualquiera o todos los ordenadores a través de la red. En un conjunto particular de realizaciones, la información puede residir en una red de área de almacenamiento ("SAN") familiar para los expertos en la técnica. De forma similar, los archivos necesarios para realizar las funciones atribuidas a los ordenadores, servidores u otros dispositivos de red se pueden almacenar localmente y/o remotamente, según corresponda. Cuando un sistema incluye dispositivos informáticos, cada uno de dichos dispositivos puede incluir elementos de hardware que pueden estar acoplados eléctricamente a través de un bus, incluyendo los elementos, por ejemplo, al menos una unidad central de procesamiento (CPU), al menos un dispositivo de entrada (por ejemplo, un ratón, teclado, controlador, pantalla táctil o panel táctil), y al menos un dispositivo de salida (por ejemplo, un dispositivo de visualización, una impresora o un altavoz). Dicho sistema puede incluir también uno o más dispositivos de almacenamiento, tales como unidades de disco, dispositivos de almacenamiento óptico y dispositivos de almacenamiento de estado

sólido tales como memoria de acceso aleatorio ("RAM") o memoria de sólo lectura (ROM), así como dispositivos de medios extraíbles, tarjetas de memoria, tarjetas flash, etc.

Tales dispositivos también pueden incluir un lector de medios de almacenamiento legible por ordenador, un dispositivo de comunicaciones (por ejemplo, un módem, una tarjeta de red (inalámbrica o cableada), un dispositivo de comunicación por infrarrojos, etc.) y una memoria de trabajo como se ha descrito anteriormente. El lector de medios de almacenamiento legible por ordenador puede estar conectado o configurado para recibir un medio de almacenamiento legible por ordenador que representa dispositivos de almacenamiento remotos, locales, fijos y/o extraíbles, así como medios de almacenamiento para contener, almacenar, transmitir y recuperar temporalmente y/o más permanentemente información legible por ordenador. El sistema y varios dispositivos también incluirán típicamente una serie de aplicaciones de software, módulos, servicios u otros elementos situados dentro de al menos un dispositivo de memoria de trabajo, incluyendo un sistema operativo y programas de aplicación, tal como una aplicación de cliente o un navegador Web. Debe apreciarse que las realizaciones alternativas pueden tener numerosas variaciones de las descritas anteriormente. Por ejemplo, también se puede utilizar hardware personalizado y/o se pueden implementar elementos particulares en hardware, software (incluido software portátil, tales como subprogramas) o ambos. Además, puede emplearse la conexión a otros dispositivos informáticos tales como dispositivos de entrada/salida de red.

Los medios de almacenamiento y medios legibles por ordenador para contener código, o porciones de código, pueden incluir cualquier medio apropiado conocido o usado en la técnica, incluyendo medios de almacenamiento y medios de comunicación, tales como, pero sin limitación, medios volátiles y no volátiles, extraíbles y no extraíbles implementados en cualquier método o tecnología para el almacenamiento y/o transmisión de información tal como instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos, incluyendo memoria RAM, ROM, EEPROM, memoria flash u otra tecnología de memoria, CD-ROM, disco versátil digital (DVD) u otro almacenamiento óptico, casetes magnéticas, cinta magnética, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que pueda usarse para almacenar la información deseada y al que pueda acceder el dispositivo del sistema. Basándose en la divulgación y en las enseñanzas proporcionadas en el presente documento, un experto en la técnica apreciará otras formas y/o métodos para implementar las diversas realizaciones.

Por consiguiente, la memoria descriptiva y los dibujos deben considerarse en un sentido ilustrativo más que restrictivo. Sin embargo, será evidente que se pueden hacer diversas modificaciones y cambios en la misma sin apartarse del alcance más amplio de la invención como se expone en las reivindicaciones.

REIVINDICACIONES

1. Un método implementado por ordenador para gestionar una instancia de base de datos replicada (234) en un entorno de base de datos (100) que utiliza un entorno de control separado, que comprende: bajo control de uno o más sistemas informáticos configurados con instrucciones ejecutables,

supervisar la información de estado para cada una de una réplica de instancia primaria y una réplica de instancia secundaria en un entorno de base de datos utilizando un componente de supervisión (218) de un entorno de control separado (208); y
en respuesta a que el componente de monitorización no puede comunicarse con una de la primera y segunda réplicas de instancia:

determinar la información de fallo que incluye si la primera y segunda réplicas de instancia son capaces de comunicarse entre sí y si la primera y segunda réplicas de instancia tienen un identificador de generación de datos común;
en base al menos en parte a la información de fallo, determinando un flujo de trabajo que se va a ejecutar en el entorno de control, incluyendo el flujo de trabajo una o más tareas a ejecutar en el entorno de base de datos en respuesta a que el componente de supervisión no pueda comunicarse con una de la primera y segunda réplicas de instancia; y
ejecutar el flujo de trabajo en el entorno de control.

2. El método implementado por ordenador de la reivindicación 1, en el que el flujo de trabajo incluye tareas para hacer que la réplica de instancia secundaria realice una operación de conmutación por error para convertirse en una nueva réplica primaria para la instancia de base de datos relacional cuando el componente de supervisión no puede comunicarse con la réplica primaria durante un periodo de tiempo mínimo, la réplica de instancia secundaria no puede comunicarse con la réplica primaria, y la segunda réplica de instancia tiene un identificador de generación de datos común como último estado conocido de la réplica primaria.

3. El método implementado por ordenador de la reivindicación 1, en el que el flujo de trabajo incluye tareas para hacer que se ejecute un proceso de recuperación de réplica de instancia secundaria que genere una nueva réplica de instancia secundaria para la instancia de base de datos relacional cuando el componente de supervisión no pueda comunicarse con la réplica secundaria durante un periodo mínimo de tiempo y la réplica de instancia primaria no pueda comunicarse con la réplica secundaria.

4. El método implementado por ordenador de la reivindicación 1, en el que el flujo de trabajo incluye tareas para almacenar información en un almacén de datos en el entorno de control sin llevar a cabo una operación de conmutación por error o recuperación cuando el componente de supervisión (218) no puede comunicarse con ninguna de la réplica primaria y la réplica secundaria durante un periodo de tiempo mínimo, las réplicas de instancia primarias y secundarias pueden comunicarse entre sí y las réplicas de instancia primarias y secundarias tienen un identificador de generación de datos común.

5. El método implementado por ordenador de la reivindicación 1, en el que la primera y segunda réplicas de instancia se aprovisionan en una única zona de datos, en zonas de datos separadas en ubicaciones geográficas separadas, en una única zona de datos a través de múltiples ubicaciones geográficas o a través de múltiples zonas de datos en una única región geográfica.

6. El método implementado por ordenador de la reivindicación 5, en el que al menos un componente de supervisión (218) está situado en una tercera zona de datos o ubicación geográfica, o en una de una primera o segunda zona de datos o ubicación geográfica.

7. El método implementado por ordenador de la reivindicación 1, en el que se proporciona al usuario un alias que permite al usuario comunicarse con una réplica de instancia primaria actual, incluyendo cuando una operación de conmutación por error hace que la réplica de instancia secundaria se convierta en una nueva réplica de instancia primaria actual.

8. El método implementado por ordenador de la reivindicación 1, en el que cada una de la primera y segunda réplicas de instancia se ejecuta en una instancia de datos independiente en el entorno de base de datos (100), estando cada instancia de datos unida a uno o más volúmenes de almacenamiento de bloques dedicados.

9. El método implementado por ordenador de la reivindicación 8, que comprende además:

replicar de forma sincrónica datos de la réplica de instancia primaria a la réplica de instancia secundaria utilizando un mecanismo de replicación a nivel de bloque operable para replicar de forma síncrona datos entre el uno o más volúmenes de almacenamiento de bloques dedicados de la primera y segunda réplicas de instancia.

10. Un sistema para gestionar una instancia de base de datos replicada (234) en un entorno de base de datos que utiliza un entorno de control separado, que comprende:

un procesador; y

un dispositivo de memoria que incluye instrucciones que, cuando son ejecutadas por el procesador, hacen que el procesador:

supervise la información de estado para cada una de una réplica de instancia primaria y una réplica de instancia secundaria en un entorno de base de datos (100) utilizando al menos un componente de supervisión (218) de un entorno de control separado (208); y

en respuesta a que el al menos un componente de monitorización no puede comunicarse con una de la primera y segunda réplicas de instancia:

determine la información de fallo que incluye si la primera y segunda réplicas de instancia son capaces de comunicarse entre sí y si la primera y segunda réplicas de instancia tienen un identificador de generación de datos común;

en base al menos en parte a la información de fallo, determine un flujo de trabajo que se va a ejecutar en el entorno de control, incluyendo el flujo de trabajo una o más tareas a ejecutar en el entorno de base de datos en respuesta a que el componente de supervisión no pueda comunicarse con una de la primera y segunda réplicas de instancia; y ejecute el flujo de trabajo en el entorno de control.

11. El sistema de la reivindicación 10, en el que el flujo de trabajo incluye tareas para hacer que la réplica de instancia secundaria realice una operación de conmutación por error para convertirse en una nueva réplica primaria para la instancia de base de datos relacional cuando el componente de supervisión no puede comunicarse con la réplica primaria durante un periodo de tiempo mínimo, la réplica de instancia secundaria no puede comunicarse con la réplica primaria, y la primera y segunda réplicas de instancia tienen un identificador de generación de datos común en un último estado conocido de la réplica primaria.

12. El sistema de la reivindicación 10, en el que el flujo de trabajo incluye tareas para hacer que se ejecute un proceso de recuperación de réplica de instancia secundaria que genere una nueva réplica de instancia secundaria para la instancia de base de datos relacional cuando el componente de supervisión no pueda comunicarse con la réplica secundaria durante un periodo mínimo de tiempo y la réplica de instancia primaria no pueda comunicarse con la réplica secundaria.

13. El sistema de la reivindicación 10, en el que el flujo de trabajo incluye tareas para almacenar información en un almacén de datos en el entorno de control sin llevar a cabo una operación de conmutación por error o recuperación cuando el componente de supervisión no puede comunicarse con ninguna de la réplica primaria y la réplica secundaria durante un periodo de tiempo mínimo, las réplicas de instancia primarias y secundarias pueden comunicarse entre sí y las réplicas de instancia primarias y secundarias tienen un identificador de generación de datos común.

14. El sistema de la reivindicación 10, en el que la primera y segunda réplicas de instancia se aprovisionan en una única zona de datos, en zonas de datos separadas en ubicaciones geográficas separadas, en una única zona de datos a través de múltiples ubicaciones geográficas o a través de múltiples zonas de datos en una única región geográfica.

15. El sistema de la reivindicación 10, en el que se proporciona al usuario un alias que permite al usuario comunicarse con una réplica de instancia primaria actual, incluyendo cuando una operación de conmutación por error hace que la réplica de instancia secundaria se convierta en una nueva réplica de instancia primaria actual.

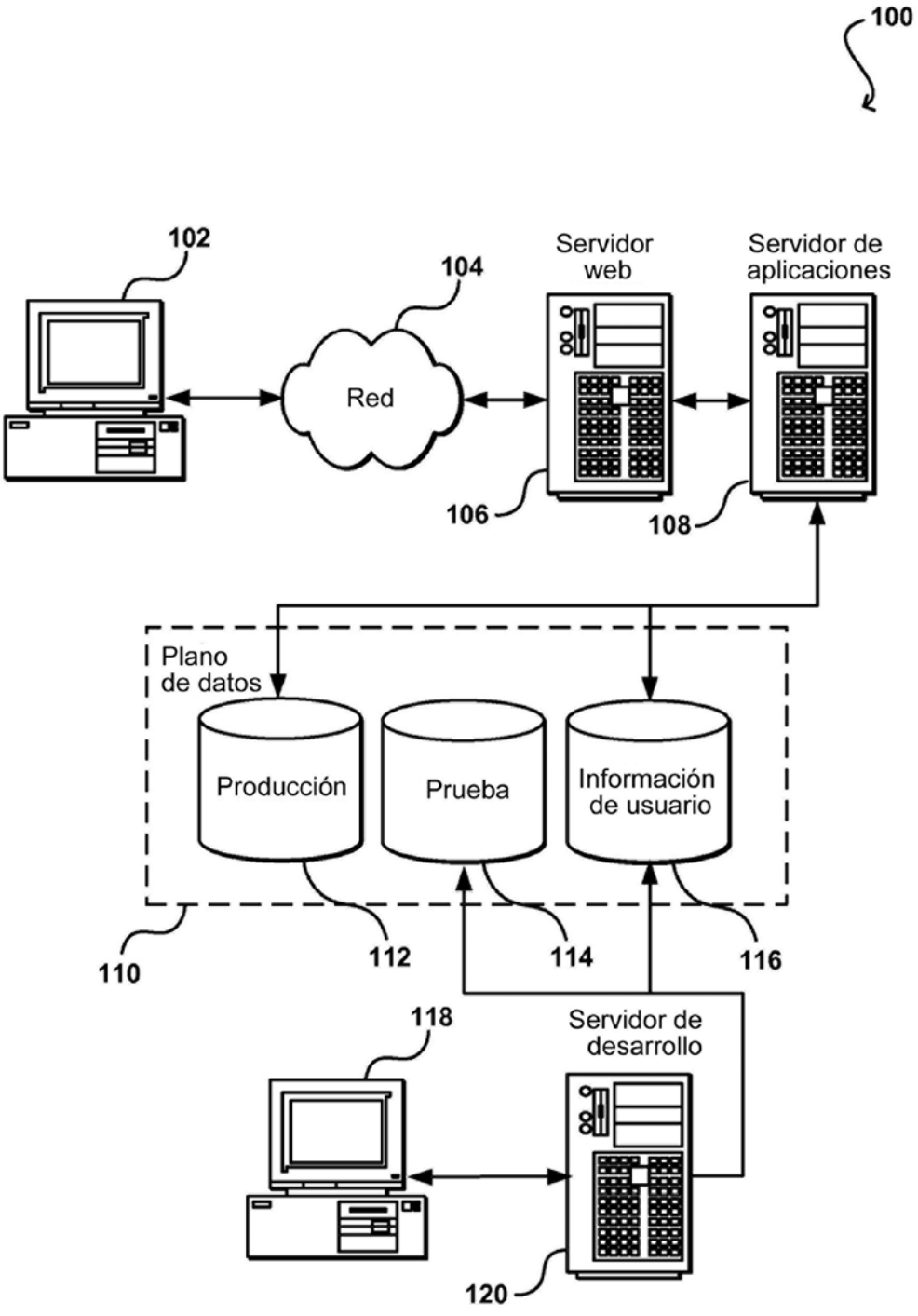
100
102
104
Servidor web
Servidor de aplicaciones
Red
106
108
Plano de datos
Producción
Prueba
Información de usuario
110
112
114
116
118
Servidor de desarrollo
120

FIG. 1

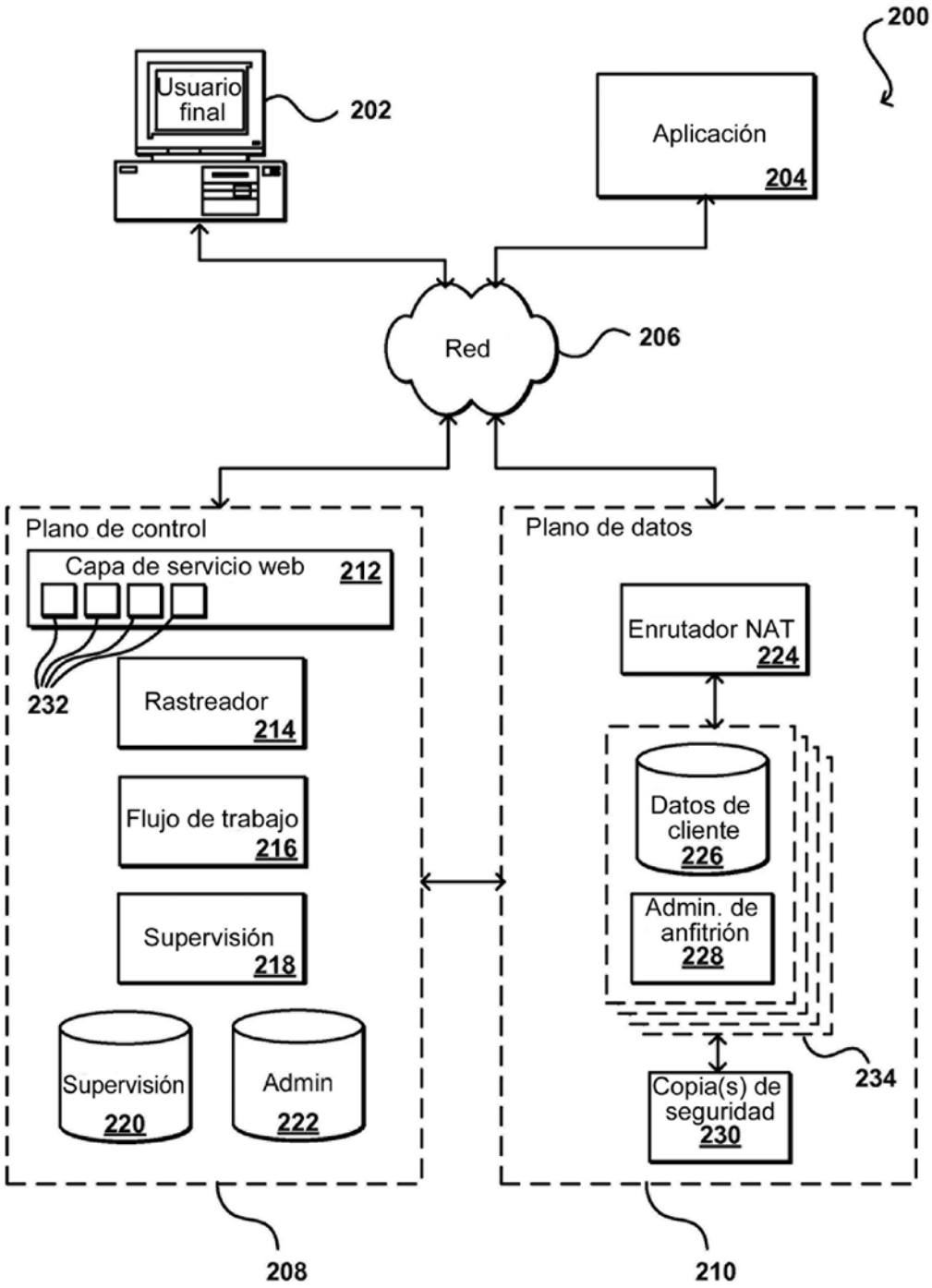
200
Usuario final
202
Aplicación
204
Red
206
Plano de control
Capa de servicio web
212
232
Rastreador
214
Flujo de trabajo
216
Supervisión
218
Supervisión
220
Admin
222
208
Plano de datos
Enrutador NAT
224
Datos de cliente
226
Admin. de anfitrión
228
234
Copia(s) de seguridad
230
210

FIG. 2

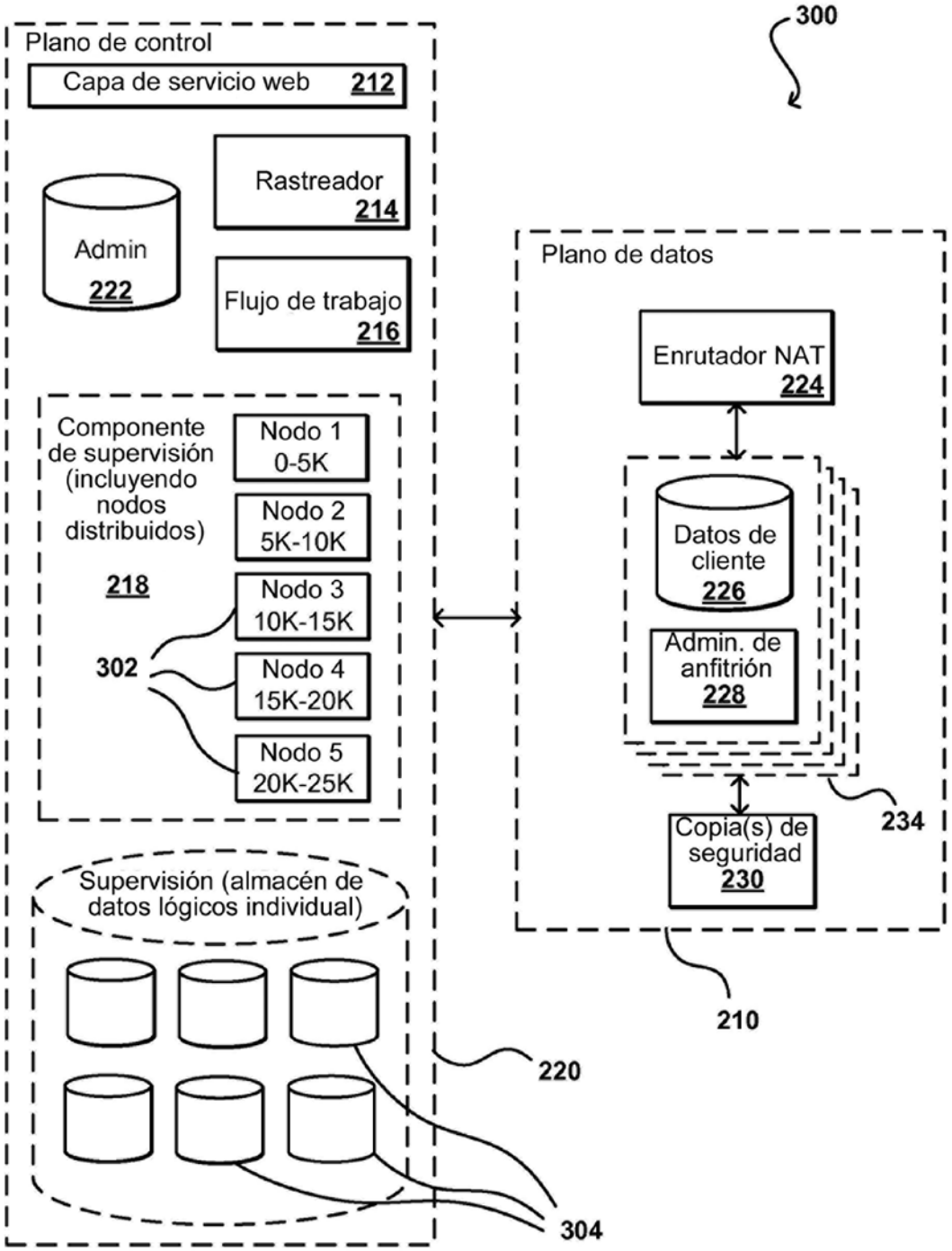
300
Plano de control
Capa de servicio web 212
Rastreador 214
Admin 222
Flujo de trabajo 216
Componente de supervisión (incluyendo nodos distribuidos) 218
Nodo 1 0-5K
Nodo 2 5K-10K
Nodo 3 10K-15K
Nodo 4 15K-20K
Nodo 5 20K-25K
302
Supervisión (almacén de datos lógicos individual)
220
304
Plano de datos
Enrutador NAT 224
Datos de cliente 226
Admin. de anfitrión 228
234
Copia(s) de seguridad 230
210

FIG. 3

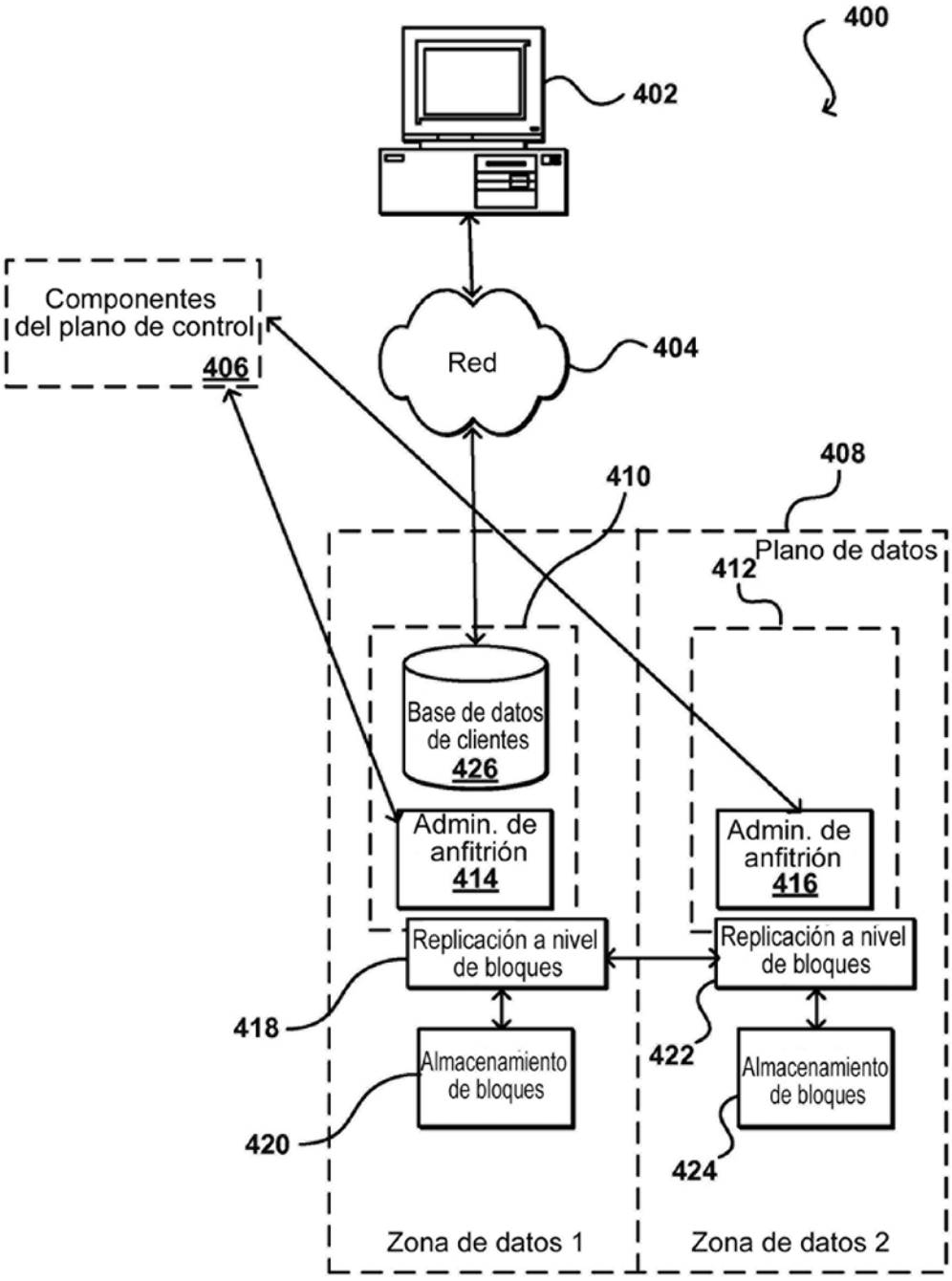
400
402
Componentes del plano de control
406
Red
404
408
410
Plano de datos
412
Base de datos de clientes
426
Admin. de anfitrión
414
Admin. de anfitrión
416
Replicación a nivel de bloques
Replicación a nivel de bloques
418
422
Almacenamiento de bloques
Almacenamiento de bloques
420
424
Zona de datos 1
Zona de datos 2

FIG. 4

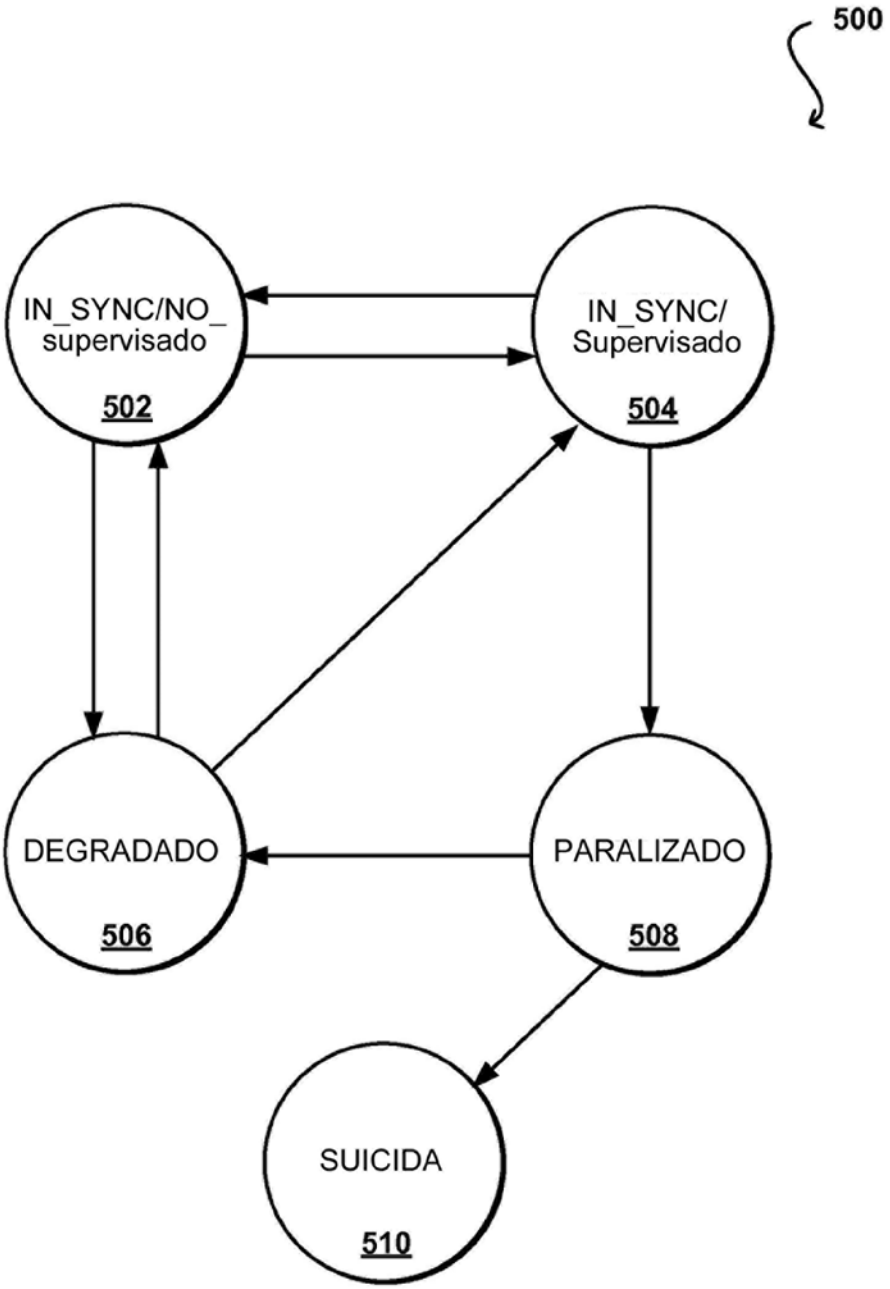
500
IN_SYNC/NO_
supervisado
502
IN_SYNC/
Supervisado
504
DEGRADADO
506
PARALIZADO
508
SUICIDA
510

FIG. 5

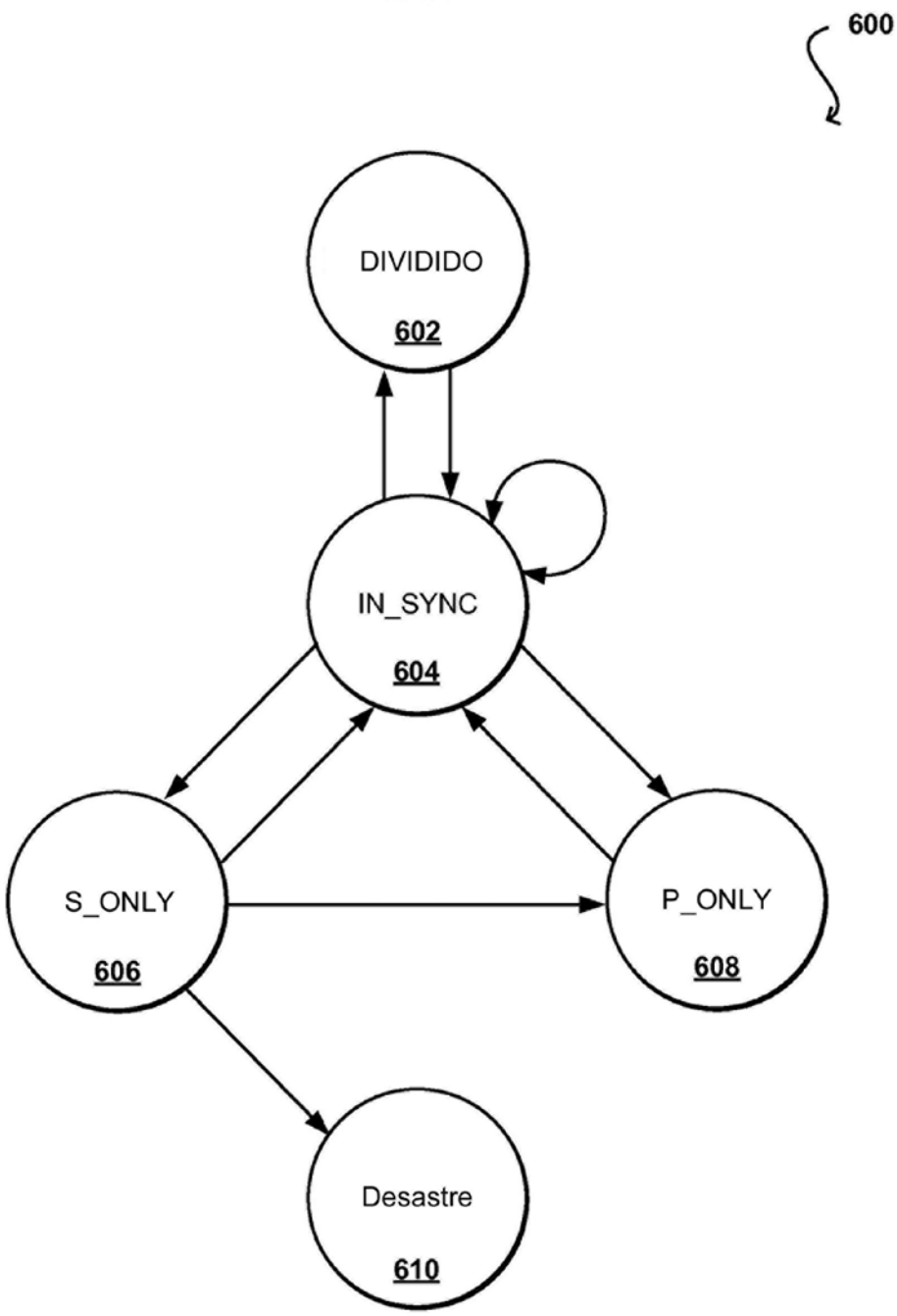
600
DIVIDIDO
602
IN_SYNC
604
S_ONLY
606
P_ONLY
608
Desastre
610

FIG. 6

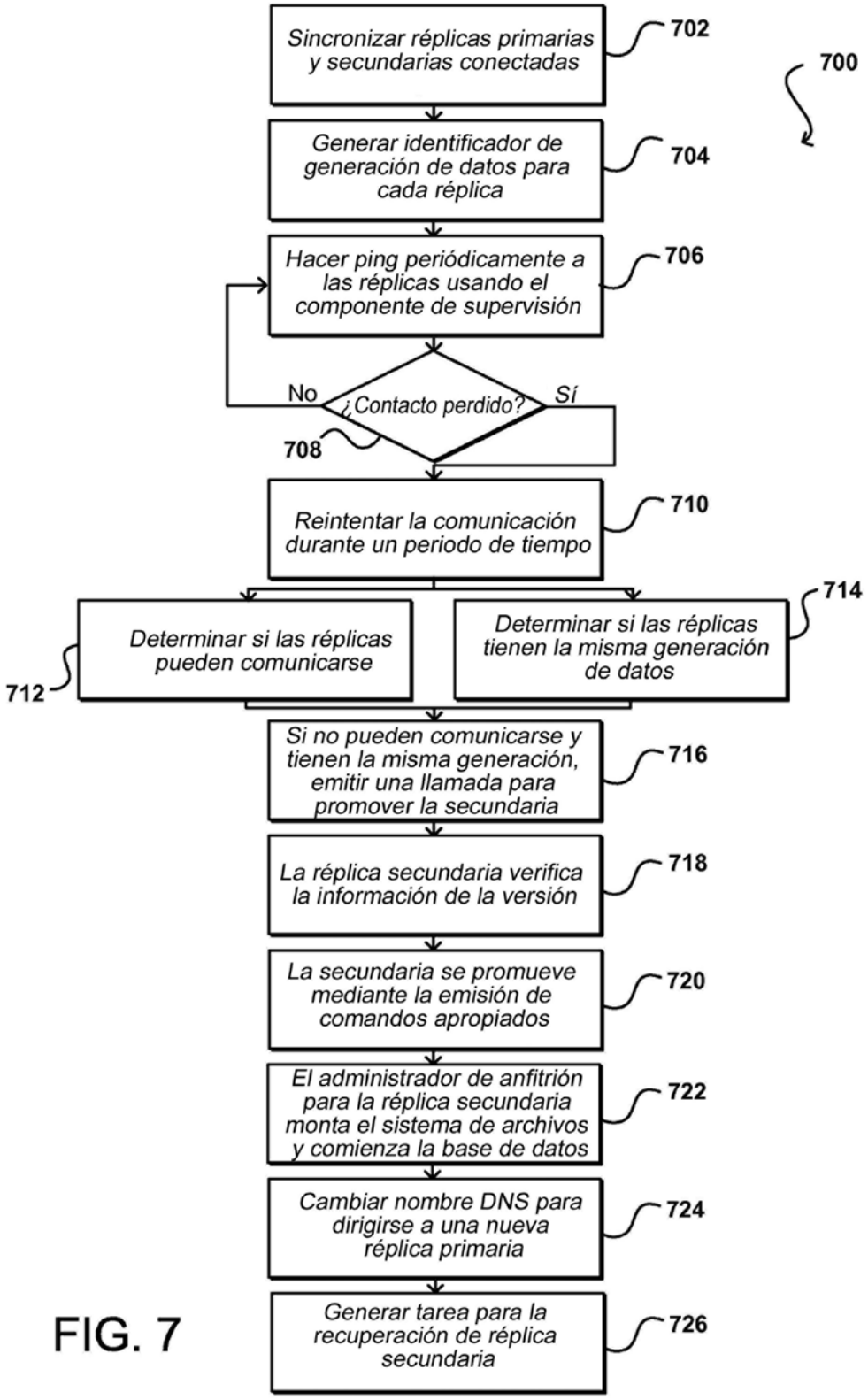
Sincronizar réplicas primarias y secundarias conectadas
702
700
Generar identificador de generación de datos para cada réplica
704
Hacer ping periódicamente a las réplicas usando el componente de supervisión
706
No
¿Contacto perdido?
Sí
708
Reintentar la comunicación durante un periodo de tiempo
710
Determinar si las réplicas pueden comunicarse
712
Determinar si las réplicas tienen la misma generación de datos
714
Si no pueden comunicarse y tienen la misma generación, emitir una llamada para promover la secundaria
716
La réplica secundaria verifica la información de la versión
718
La secundaria se promueve mediante la emisión de comandos apropiados
720
El administrador de anfitrión para la réplica secundaria monta el sistema de archivos y comienza la base de datos
722
Cambiar nombre DNS para dirigirse a una nueva réplica primaria
724
Generar tarea para la recuperación de réplica secundaria
726

FIG. 7

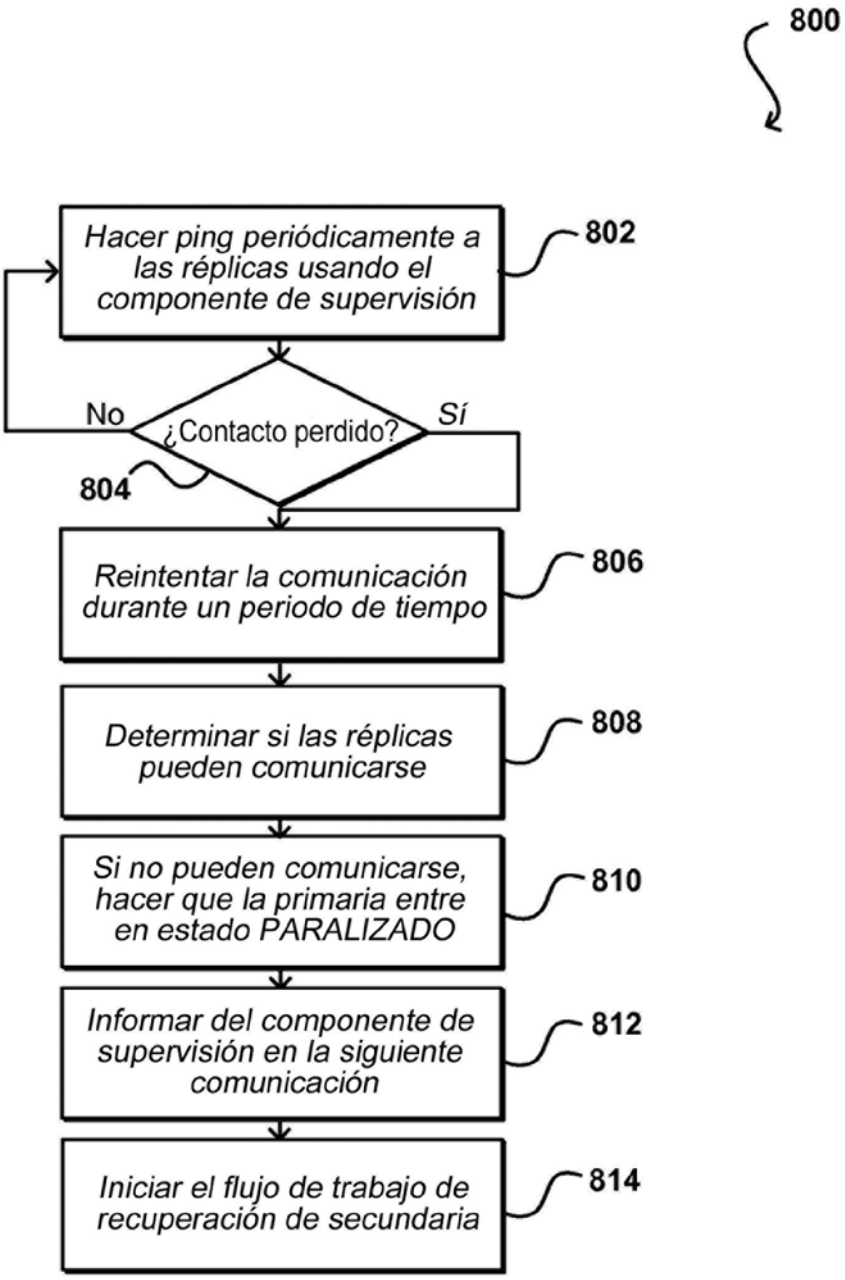
800
Hacer ping periódicamente a las réplicas usando el componente de supervisión
802
No
¿Contacto perdido?
Sí
804
Reintentar la comunicación durante un periodo de tiempo
806
Determinar si las réplicas pueden comunicarse
808
Si no pueden comunicarse, hacer que la primaria entre en estado PARALIZADO
810
Informar del componente de supervisión en la siguiente comunicación
812
Iniciar el flujo de trabajo de recuperación de secundaria
814

FIG. 8

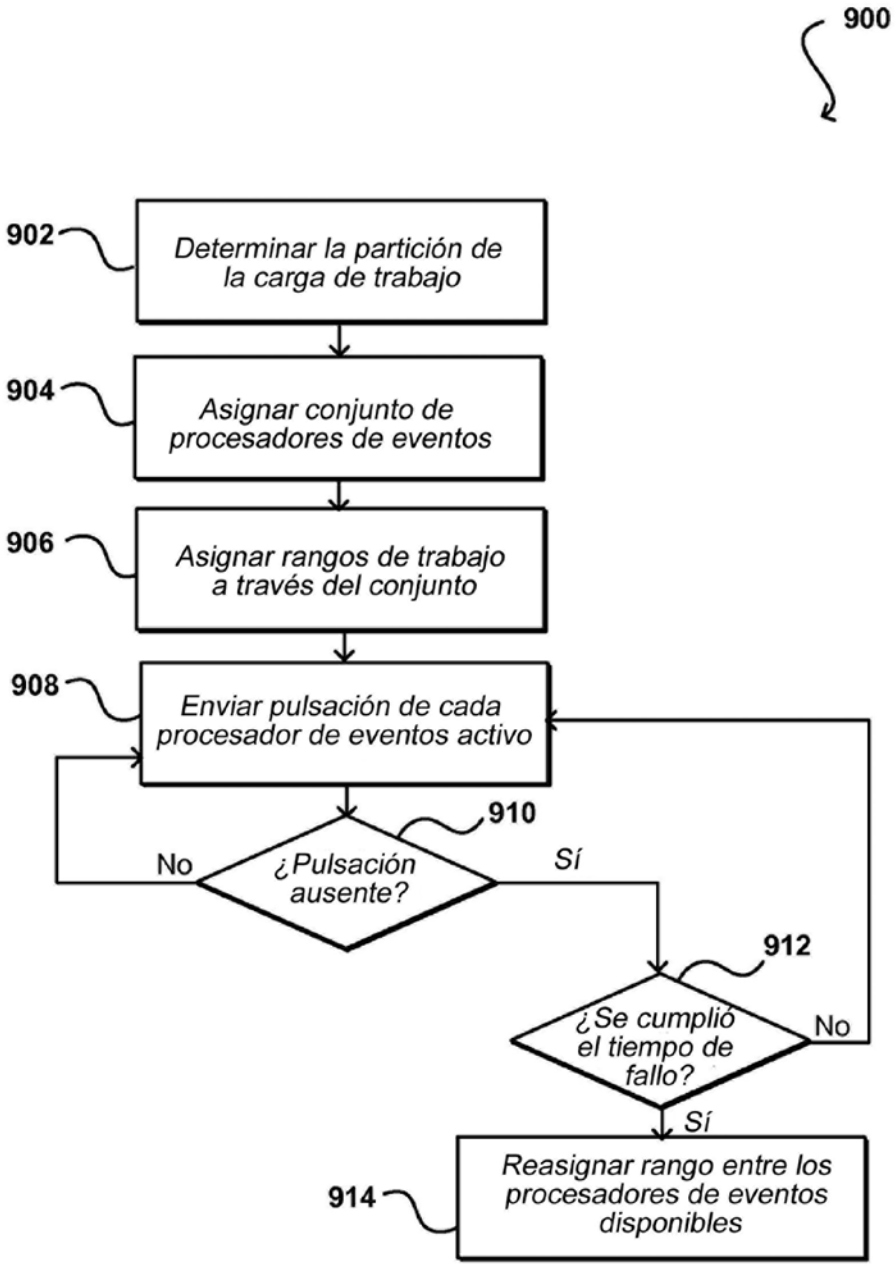
900
902
Determinar la partición de la carga de trabajo
904
Asignar conjunto de procesadores de eventos
906
Asignar rangos de trabajo a través del conjunto
908
Enviar pulsación de cada procesador de eventos activo
910
¿Pulsación ausente?
No
Sí
912
¿Se cumplió el tiempo de fallo?
No
Sí
914
Reasignar rango entre los procesadores de eventos disponibles

FIG. 9

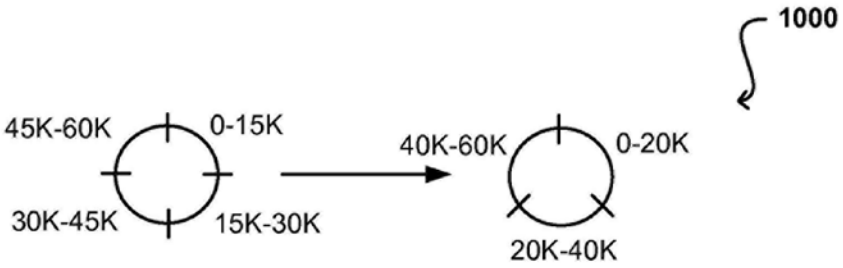
1000
45K-60K
0-15K
30K-45K
15K-30K
40K-60K
0-20K
20K-40K

FIG. 10

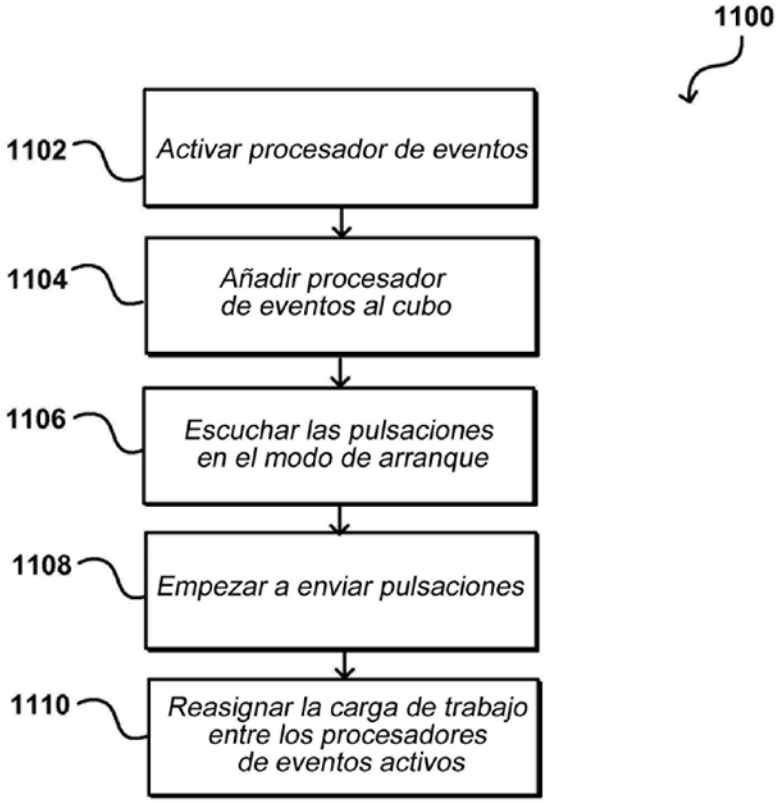
1100
1102
Activar procesador de eventos
1104
Añadir procesador de eventos al cubo
1106
Escuchar las pulsaciones en el modo de arranque
1108
Empezar a enviar pulsaciones
1110
Reasignar la carga de trabajo entre los procesadores de eventos activos

FIG. 11