(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 723 781**

(51) Int. CI.:

G06F 17/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **03.08.2011 E 11306011 (5)**

(97) Fecha y número de publicación de la concesión europea: **06.02.2019 EP 2555129**

(54) Título: **Método y sistema para mantener consistencia fuerte de contenidos replicados distribuidos en un sistema de cliente/servidor**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
02.09.2019

(73) Titular/es:

AMADEUS S.A.S. (100.0%)
485 Route du Pin Montard, Sophia Antipolis
06410 Biot, FR

(72) Inventor/es:

TOUFFAIT, GUILLAUME;
AMAR, VIRGINIE;
LAFONT, CAROLINE;
DEFAYET, CHRISTOPHE y
COLLENDAVELLOO, YAN

(74) Agente/Representante:

SUGRAÑES MOLINÉ, Pedro

Observaciones:

Véase nota informativa (Remarks, Remarques o Bemerkungen) en el folleto original publicado por la Oficina Europea de Patentes

DESCRIPCIÓN

Método y sistema para mantener consistencia fuerte de contenidos replicados distribuidos en un sistema de cliente/servidor

Campo de la invención

La presente invención se refiere en general a sistemas de procesamiento de datos y en particular a arquitecturas de software de cliente/servidor distribuido. Aún más específicamente, la presente invención se refiere a un método y a un sistema para mantener consistencia entre contenidos de ficheros de caché distribuidos a través de una pluralidad de nodos de procesamiento, mientras que se asegura su disponibilidad en casi tiempo real.

Antecedentes de la invención

El modelo cliente/servidor que ha emergido a finales de los 80 es una arquitectura versátil y modular que se ideó para mejorar usabilidad, flexibilidad, interoperabilidad y escalabilidad en comparación con la informática de compartición de tiempo, de ordenadores centrales centralizada que era la norma en ese momento. La arquitectura cliente/servidor desde entonces ha sustituido por completo de manera progresiva las arquitecturas de software de ordenador central anteriores donde toda la inteligencia estaba dentro del ordenador de anfitrión central y donde los usuarios interactuaban con el anfitrión a través de terminales tontos. Sin embargo, si los ordenadores centrales aún están en uso, es únicamente como servidores potentes en diversas arquitecturas cliente/servidor donde los terminales tontos también se han sustituido por interfaces de usuario gráficas (GUI) inteligentes que pueden realizar auto procesamiento de los datos recibidos y transmitidos desde/hasta los servidores.

En sistemas de procesamiento de datos modernos, una arquitectura cliente/servidor ampliamente en uso y que puede soportar un gran número de clientes localizados de manera remota es la denominada arquitectura de nivel 3. Un ejemplo de tal arquitectura se ilustra en la Figura 1. El nivel maestro 100 está creado de manera tradicional alrededor de un sistema de base de datos 120, posiblemente un repositorio grande o muy grande de todos los datos necesarios para la operación diaria de cualquier organización empresarial, compañía o empresa para realizar toda clase de operaciones comerciales y administrativas. La base de datos es en su mayoría del tipo relacional, es decir, está bajo el control de un sistema de gestión de base de datos relacional o RDBMS. Está administrada típicamente a través de uno o más servidores maestros 112 por administradores del sistema de procesamiento de datos de las GUI 140. Los administradores son en general los únicos usuarios del sistema autorizado para actualizar directamente los contenidos de la base de datos.

El nivel intermedio o medio del sistema de 3 niveles ejemplar de la Figura 1 es el nivel de aplicación 200 desde donde se ejecutan todas las aplicaciones de software 240 específicas de la organización, el propietario del sistema de procesamiento de datos. Esta colección de aplicaciones específicas, a menudo denominadas de manera global como el software de soporte intermedio, es el software propietario de la organización. Se usa para servir a todos los clientes remotos de la organización desde su repositorio de datos 120 a través de los servidores maestros 110. Los clientes remotos forman el tercer nivel 300 de la arquitectura de 3 niveles. Las consultas del nivel de cliente 300 se procesan de esta manera y se responden por las aplicaciones específicas del nivel intermedio 200 en datos capturados desde el nivel maestro 100.

En una arquitectura de nivel 3, cuando necesita servirse un número mayor de clientes remotos, se obtiene escalabilidad del sistema para mantener rendimientos globales añadiendo nodos de procesamiento independientes en el nivel medio para aumentar la potencia de procesamiento global del sistema de procesamiento de datos. Por lo tanto, el nivel de aplicación 200 está comprendido en general de varios nodos de procesamiento independientes que se denominan, en la siguiente descripción, como nodos esclavos 210. A continuación, una práctica común para evitar que el nivel maestro 100 se desborde por demasiadas solicitudes de datos de un número creciente de nodos esclavos, es tener los procesos aplicativos 240 funcionando en piezas de datos llevadas desde la base de datos maestra y almacenadas en cada nodo de aplicación siempre que sea necesario. En el sistema ejemplar de la Figura 1 esto toma la forma de ficheros de caché 250 en los que los procesos aplicativos 240 pueden trabajar sin tener que incurrir en retardos largos para conseguirlos de la base de datos maestra a través de los servidores maestros cada vez que son necesarios. En un sistema de procesamiento de datos de este tipo la potencia de procesamiento y las aplicaciones de software están por lo tanto distribuidas, es decir, particionadas, en tantos nodos 210 como sean necesarios para alcanzar el nivel de potencia de procesamiento necesario para servir a todos los clientes remotos 300 del sistema. Por lo que son los ficheros de caché 250 distribuidos.

En un entorno informático distribuido de este tipo, se ha probado, sin embargo, que algunas propiedades deseables de un sistema de datos distribuido no pueden garantizarse todas de manera simultánea. Como se ilustra en la Figura 2 estas propiedades esperadas 40 de un sistema de procesamiento de datos distribuido son: consistencia, disponibilidad y escalabilidad. Un teorema conocido como el teorema de CAP, establece que un sistema distribuido puede satisfacer cualesquiera dos de estas propiedades al mismo tiempo pero no todas las tres. CAP, que significa: consistencia, disponibilidad y tolerancia de partición; se ha conjeturado por primera vez en 2000 por E. Brewer, Profesor en la Universidad de California, Berkeley, Estados Unidos. Una demostración del teorema se ha realizado

más tarde en un artículo escrito por N. Lynch y S. Gilbert, publicado en 2002 en ACM SIGACT News, v.33 edición 2, páginas 51-59. La propiedad de tolerancia de partición de CAP está estrechamente vinculada a escalabilidad puesto que, como se ha analizado anteriormente, el procesamiento de potencia global del sistema realmente se obtiene al distribuir, es decir, particionándolo a través de nodos de procesamiento independientes.

La consistencia y disponibilidad 41 pueden cumplirse completamente en arquitecturas de 3 niveles únicamente si los datos usados por las aplicaciones de nivel medio provienen de la base de datos maestra. Esto puede obtenerse como el gasto de generar un tráfico descendente muy alto desde el nivel maestro 100 al nivel de aplicación 200 simplemente para contestar consultas del nivel de cliente 300 dando como resultado también una ocupación muy alta de la base de datos maestra para contestarlo. Esto entra en conflicto con la administración y actualización de la base de datos maestra por usuarios administrativos (140) incluso aunque la proporción de escrituras en la base de datos sea relativamente baja en general. El acceso a la base de datos y el tráfico en la red entre niveles de datos y aplicación son de manera evidente cuellos de botella que limitan los rendimientos cuando el número de usuarios del nivel de cliente aumenta.

La disponibilidad y escalabilidad 42 pueden conseguirse en una arquitectura de 3 niveles como la ejemplar mostrada en la Figura 1 teniendo distribuidos ficheros de caché 250 para superar los problemas anteriores de ocupación de base de datos y alto tráfico entre niveles de datos y aplicaciones. Sin embargo, en este caso, no hay garantía de que los contenidos de fichero de caché son consistentes entre nodos esclavos y con contenidos de la base de datos maestra puesto que están distribuidos a través de nodos informáticos independientes.

Es por lo tanto un objeto de la invención proporcionar una solución a este problema. En una arquitectura cliente/servidor de nivel 3 donde las aplicaciones de cliente y ficheros replicados están distribuidos a través de una pluralidad de nodos esclavos independientes, la invención desvela un método y un sistema para mantener consistencia fuerte entre contenidos de fichero replicados y disponibilidad total mientras se conserva alguna escalabilidad 43.

Objetos, características y ventajas adicionales de la presente invención se harán evidentes para los expertos en la materia tras la examinación de la siguiente descripción en referencia a los dibujos adjuntos. Se pretende que puedan incorporarse cualesquiera ventajas adicionales en el presente documento.

El documento US2003/0023898A1 desvela un método de dos fases para replicación de datos mediante el cual para cometer la actualización, el servidor maestro debe recibir acuse de recibo de procesamiento satisfactorio de cada uno de los esclavos, de otra manera se cancela la actualización.

Sumario de la invención

De acuerdo con un aspecto, la invención describe un método de mantenimiento de la consistencia de ficheros replicados distribuidos a través de una pluralidad de parte de nodos esclavos de procesamiento independientes de un nivel de aplicación de un sistema de procesamiento de datos de cliente/servidor de múltiples clientes, estando distribuidos los ficheros replicados desde un nodo maestro de un nivel maestro. El método comprende las siguientes etapas realizadas con al menos un procesador:

- recibir una solicitud de actualización en al menos un servidor maestro del nivel maestro para actualizar una base de datos maestra del sistema de procesamiento de datos;
- basándose en dicha actualización que genera y almacena una nueva versión de un fichero replicado almacenado en un sistema de ficheros compartido del nivel maestro;
- proporcionar, una notificación de disponibilidad de la nueva versión del fichero replicado a todos los nodos esclavos;
- en cada nodo esclavo, iniciar la precarga del sistema de ficheros compartido de la nueva versión del fichero replicado y tras la finalización de la precarga, realizar acuse de recibo de la finalización satisfactoria de la precarga;

• únicamente si todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga a continuación realizar las siguientes etapas:

  - recibir en el servidor maestro una notificación de consecución de precarga;
  - desde el servidor maestro, actualizar la base de datos maestra con datos de la nueva versión del fichero replicado haciéndose por lo tanto el uso de la nueva versión del fichero replicado por el nivel maestro;
  - hacer el uso de la nueva versión del fichero replicado en una base de datos esclava del nivel de aplicación dispuesto para mantener registro de todas las versiones de fichero replicado;
  - reenviar al servidor maestro una notificación del compromiso de todos los nodos esclavos;
  - desde el servidor maestro, realizar acuse de recibo de consecución de la actualización en respuesta a la solicitud de actualización recibida,

• si no todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga, a continuación recibir en el nodo maestro una notificación de error sin actualizar la base de datos maestra y hacer el uso de la nueva versión en la base de datos esclava.

Por lo tanto, la invención proporciona una solución eficaz para responder a una consulta de información en casi tiempo real mientras se suprime una consistencia fuerte a través de un número de nodos posiblemente alto que tiene la pieza de información requerida para responder a la consulta. Por lo tanto, la invención posibilita el mantenimiento de consistencia fuerte, disponibilidad con una latencia muy corta y escalabilidad mejorada.

Opcionalmente, la invención puede comprender al menos una de las siguientes características opcionales:
En la presente invención, lo que se designa como ficheros replicados son ficheros completos que se replican desde la base maestra y se llevan como tal en los nodos esclavos para expedir el procesamiento de los datos correspondientes por los procesos aplicativos. No son de por sí piezas de datos pequeñas llevadas desde una memoria de extremo trasero más lenta en una memoria caché. En el contexto de los ficheros replicados de la invención, también denominados como réplica son de hecho ficheros completos en los que un proceso aplicativo en un nodo esclavo puede funcionar sin experimentar ninguna pérdida de datos para completar.

De acuerdo con una realización particular pero no limitativa, los ficheros replicados, también denominados como réplica, son ficheros de caché.

La etapa de proporcionar una notificación de disponibilidad de la nueva versión del fichero replicado a todos los nodos esclavos comprende las siguientes etapas:

- reenviar desde el servidor maestro una notificación de disponibilidad de la nueva versión del fichero replicado a un nodo esclavo de sincronización tomado entre los nodos esclavos;
- difundir desde el nodo esclavo de sincronización dicha notificación de disponibilidad a todos los otros nodos esclavos.

La etapa de realizar acuse de recibo de la finalización satisfactoria de la precarga comprende una etapa en la que los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga a un proceso de servidor del nodo esclavo de sincronización.

Una vez que todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga al proceso de servidor del nodo esclavo de sincronización y antes de la etapa de recepción en el servidor maestro de una notificación de consecución de precarga, el nodo esclavo de sincronización reenvía al servidor maestro dicha notificación de consecución de precarga.

Tras la finalización de la etapa de hacer el uso de la nueva versión del fichero replicado por el nivel maestro, el nodo maestro reenvía una notificación de compromiso al nodo esclavo de sincronización y a continuación el nodo esclavo de sincronización activa la etapa de hacer el uso de la nueva versión del fichero replicado en la base de datos esclava.

En el comienzo de una transacción iniciada por una consulta para información de un usuario, al menos en cada nodo esclavo implicado en dicha transacción de un proceso aplicativo interroga la base de datos esclava y se da instrucciones de si conmutar desde una versión actual del fichero replicado a la nueva versión precargada del fichero replicado.

Preferentemente, en el comienzo de una transacción iniciada por una consulta para información de un usuario, en cada nodo esclavo implicado en dicha transacción de un proceso aplicativo lee una etiqueta de una versión actual del fichero replicado, indicando dicha etiqueta si está disponible una nueva versión del fichero replicado. Si la etiqueta indica que está disponible una nueva versión del fichero replicado, a continuación el proceso aplicativo interroga la base de datos esclava y se da instrucciones sobre si conmutar de la versión actual del fichero replicado a la nueva versión del fichero replicado.

La etiqueta se establece a finalización satisfactoria de precarga de una nueva versión del fichero replicado y se resetea leyendo en primer lugar por el proceso aplicativo.

La solicitud de actualización recibida comprende al menos una actualización y las etapas de generación y almacenamiento de una nueva versión de un fichero replicado en el sistema de ficheros compartido son un proceso incremental que incluye, en el servidor maestro, las etapas de:

recuperar del sistema de ficheros compartido una versión actual de un fichero replicado y convertirlo a un formato apropiado de la versión actual del fichero replicado a actualizarse;

aplicar la actualización sobre la versión actual y convertir y almacenar la nueva versión del fichero replicado en el sistema de ficheros compartido.

Los ficheros replicados se comprimen cuando se almacenan en el sistema de ficheros compartido.

Los ficheros replicados almacenados en el sistema de ficheros compartido se descomprimen y convierten en un formato apropiado por el servidor maestro cuando se realiza la actualización incremental y por los nodos esclavos tras

la precarga.

Si no todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga, a continuación el nodo maestro envía una notificación de error a un administrador que envió dicha solicitud de actualización.

La base de datos maestra puede almacenarse en una única máquina o en una pluralidad de máquinas del nivel maestro. La base de datos esclava puede almacenarse en una única máquina o en una pluralidad de máquinas del nivel de aplicación.

El sistema de ficheros compartido es un almacenamiento de conexión de red NAS.

De acuerdo con otro aspecto, la invención se refiere a un producto de programa informático almacenado en un medio de memoria legible por ordenador no transitorio y que realiza el método de acuerdo con una cualquiera de las características anteriores.

De acuerdo con otro aspecto, la invención se refiere a un método de mantenimiento de la consistencia de ficheros replicados que contienen reglas de empresa, estando distribuidos dichos ficheros replicados a través de una pluralidad de nodos esclavos de procesamiento independientes formando una parte de un nivel de aplicación de un sistema de procesamiento de datos de cliente/servidor de múltiples niveles, estando distribuidos los ficheros replicados desde un nodo maestro de un nivel maestro, en el que el método comprende las siguientes etapas realizadas con al menos un procesador de datos:

- recibir una solicitud de actualización en al menos un servidor maestro del nivel maestro para actualizar reglas de negocio almacenadas en una base de datos maestra del sistema de procesamiento de datos;
- basándose en dicha actualización que genera y almacena una nueva versión de un fichero replicado almacenado en un sistema de ficheros compartido del nivel maestro;
- proporcionar una notificación de disponibilidad de la nueva versión del fichero replicado a todos los nodos esclavos;
- en cada nodo esclavo, iniciar la precarga del sistema de ficheros compartido de la nueva versión del fichero replicado y tras la finalización de la precarga, realizar acuse de recibo de la finalización satisfactoria de la precarga;

• únicamente si todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga a continuación realizar las siguientes etapas:

  - recibir en el servidor maestro una notificación de consecución de precarga;
  - desde el servidor maestro, actualizar la base de datos maestra con datos de la nueva versión del fichero replicado haciéndose por lo tanto el uso de la nueva versión del fichero replicado por el nivel maestro;

si no todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga, recibir a continuación en el nodo maestro una notificación de error.

Opcionalmente, la invención puede comprender al menos una de las siguientes características opcionales.

Ventajosamente, cada regla de negocio comprende un conjunto de criterios y un contenido. Preferentemente, cada criterio está asociado a un peso, permitiendo de esta manera un motor de búsqueda para identificar la regla más relevante cuando se lanza una búsqueda.

El conjunto de criterios comprende al menos uno de: un punto de venta, un país, un grupo de países, un país de origen de un vuelo, un destino de un vuelo, un perfil de cliente.

El sistema de procesamiento de datos de cliente/servidor de múltiples niveles es parte de un inventario de un proveedor de viajes.

Más en general, los nodos esclavos están conectados a al menos uno de: un inventario de una línea aérea, un sistema de control de salidas de una línea aérea, un sistema de gestión de ingresos de una línea aérea, un sistema de contabilidad de ingresos de una aerolínea, un servidor de tiques electrónicos de una línea aérea.

Otro aspecto de la invención es un sistema de procesamiento de datos de cliente/servidor de múltiples niveles distribuido que comprende un nivel de aplicación y un nivel maestro, comprendiendo el nivel de aplicación una pluralidad de nodos esclavos de procesamiento independientes. El nivel maestro comprende un nodo maestro dispuesto para distribuir ficheros replicados a los nodos esclavos, una base de datos maestra y un servidor maestro. El nivel maestro comprende un sistema de ficheros compartido. El servidor maestro está dispuesto para recibir una solicitud de actualización para actualizar la base de datos maestra, para generar una nueva versión de un fichero replicado almacenado en el sistema de ficheros compartido y para almacenar dicha nueva versión en el sistema de ficheros compartido. El sistema de acuerdo con la invención también comprende una base de datos esclava conectada a todos los nodos esclavos y dispuesta para mantener registro de todas las versiones de fichero replicado. El sistema está configurado para:

- proporcionar una notificación de disponibilidad de la nueva versión del fichero replicado a todos los nodos esclavos;
- en cada nodo esclavo, iniciar la precarga desde el sistema de ficheros compartido de la nueva versión del fichero replicado y tras la finalización de la precarga, realizar acuse de recibo de la finalización satisfactoria de la precarga;
- únicamente si todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga a continuación realizar las siguientes etapas: recibir en el servidor maestro una notificación de consecución de precarga; desde el servidor maestro, actualizar la base de datos maestra con datos de la nueva versión del fichero replicado haciendo por lo tanto el uso de la nueva versión del fichero replicado por el nivel maestro; hacer el uso de la nueva versión del fichero replicado en la base de datos esclava; reenviar al servidor maestro una notificación de compromiso de todos los nodos esclavos; desde el servidor maestro, realizar acuse de recibo de la consecución de la actualización en respuesta a la solicitud de actualización recibida,
- si no todos los nodos esclavos realizan acuse de recibo de la finalización satisfactoria de la precarga, a continuación recibir en el nodo maestro una notificación de error sin actualizar la base de datos maestra y sin hacer el uso de la nueva versión en la base de datos esclava.

Breve descripción de los dibujos

La FIGURA 1 ilustra un ejemplo de un sistema de procesamiento de datos de 3 niveles distribuido convencional.
La FIGURA 2 analiza consistencia, disponibilidad y escalabilidad de sistemas de procesamiento de datos distribuidos.
La FIGURA 3 ilustra un sistema de procesamiento de datos ejemplar de acuerdo con la invención en el que los contenidos de una base de datos maestra se distribuyen en nodos esclavos en forma de ficheros replicados.
La FIGURA 4, que está comprendida de las Figuras 4a y 4b, describe las etapas del método para conseguir consistencia fuerte entre los ficheros replicados distribuidos de un sistema de procesamiento de datos de acuerdo con la invención.
La FIGURA 5 describe la actualización incremental que se realiza cuando se crea una nueva versión (Vn+1) de un fichero replicado.

Descripción detallada

La siguiente descripción detallada de la invención se refiere los dibujos adjuntos. Aunque la descripción incluye realizaciones ejemplares, son posibles otras realizaciones, y pueden realizarse cambios a las realizaciones descritas sin alejarse del espíritu y alcance de la invención.

La **Figura 3** ilustra un sistema de procesamiento de datos distribuido ejemplar de acuerdo con la invención que comprende un nivel de aplicación 200 y un nivel maestro 100 también denominados como el nivel de datos 100. El nivel de aplicación 200 comprende una pluralidad de nodos esclavos de procesamiento independientes 210, 210', 210". El nivel maestro 100 comprende un nodo maestro 110, una base de datos maestra 120 y un servidor maestro 112. Los contenidos de la base de datos maestra 120 se distribuyen en nodos esclavos 210, 210', 210" en forma de ficheros replicados 250, 150 de modo que las piezas de datos correspondientes pueden usarse por aplicaciones de cliente 240, 240', 240" que se ejecutan en estos nodos esclavos 210, 210', 210" sin tener que interrogar la base de datos maestra 120.

El nivel maestro 100 incluye un nodo maestro 110 comprendido de uno o más servidores 112. El nodo maestro 100 está a cargo de gestionar el repositorio de datos del sistema, es decir, la base de datos maestra 120, en general un sistema de base de datos grande o muy grande que mantiene todos los datos necesarios para realizar cualquier tipo de actividades empresariales grandes que incluyen aplicaciones comerciales e industriales y que cubren posiblemente también toda clase de actividades administrativas, educacionales y gubernamentales también. En la industria de la línea aérea y de viajes esto puede ser, por ejemplo, un sistema de distribución global o GDS. Un GDS es cualquier sistema de procesamiento de datos grande puesto en su lugar por compañías de la industria de viajes para permitir acceso en tiempo real a tarifas de línea aérea, planificaciones y disponibilidad de asiento y en general a cualquier producto de viaje. Los GDS ofrecen la capacidad de que agencias de viaje e individuos, a través de numerosos sitios de viajes en línea, reserven reservas y generen tiques de viaje a lo largo de todo el mundo.

Una base de datos 120 de este tipo se administra a través del servidor maestro 112 por cualesquiera usuarios autorizados de una interfaz de usuario 140 del sistema, en general una interfaz de usuario gráfica (GUI). Los usuarios administrativos de la base de datos son aquellos que están permitidos a actualizar sus contenidos. Por ejemplo, en el caso de un GDS, para añadir o modificar reglas de negocio que se aplican a la reserva de tiques de viaje o para modificar ofertas de producto de viaje. Esto consiste en su mayoría en este caso de actualizaciones llevadas a una oferta actual de productos de viaje. Las actualizaciones deben por lo tanto replicarse de manera ordenada en los ficheros replicados distribuidos usados por aplicaciones de cliente 240, 240', 240" que se ejecutan en los nodos esclavos 210, 210', 210".

El nivel maestro 100 también incluye un sistema en ficheros compartido 160, por ejemplo un sistema de almacenamiento de conexión en red o NAS 160 que tiene como objetivo mantener las nuevas versiones de los ficheros replicados que se generan tras solicitud de un usuario administrativo. El uso de NAS 160 y de otros componentes del sistema se explica adicionalmente en la Figura 4 que analiza el método de mantener coherentes los diversos ficheros

replicados distribuidos de los nodos esclavos 210, 210', 210" con los contenidos de la base de datos maestra 120.

En lo que respecta al nivel de aplicación de cliente 200 está comprendido de una pluralidad de nodos esclavos 210, 210', 210" cada uno puede servir suficientes usuarios finales remotos que constituyen el tercer nivel del sistema (no mostrado en la Figura 3). En el ejemplo anterior de un GDS estos usuarios finales localizados de manera remota incluyen, por ejemplo, agentes de viaje de agencias de viajes convencionales y usuarios individuales de sitios de viaje en línea que se conectan a través de una red pública a estos sitios, es decir: a través de la Internet. Muchas aplicaciones de cliente de software diferentes que activan los procesos aplicativos 240, 240', 240" de los nodos esclavos 210, 210', 210" pueden ejecutarse por lo tanto a partir de los ficheros replicados que se llevan en una memoria compartida 230, 230', 230" de cada nodo esclavo 210, 210', 210". Por lo tanto, los datos necesarios por estas aplicaciones son fácilmente accesibles por cualquiera de los procesos aplicativos y no necesitan duplicarse para cada uno de ellos reduciendo drásticamente el uso de memoria. El rango y número de productos de software aplicativos ejecutados en los nodos esclavos 210, 210', 210" es potencialmente muy amplio y en la práctica simplemente limitado por los recursos informáticos de los nodos. Es en gran medida dependiente del tipo de negocio realizado por el propietario del sistema de procesamiento de datos. En la industria de viajes, para un GDS, esto incluye aplicaciones específicas como los servicios de reserva electrónica y generación de tiques ofrecidos ahora para todos los viajeros y el control de salida de pasajeros en los aeropuertos. Aplicaciones más en común a cualquier tipo de negocio incluyen aquellas con respecto a la disponibilidad de bienes y servicios, gestión de ingresos y contabilidad de ingresos. Por lo tanto, los usuarios del nivel de aplicación 200 no están limitados a ser clientes sino que también incluyen algunos profesionales a cargo de los bienes de la compañía que posee el sistema de procesamiento de datos.

El nivel de aplicación de cliente 200 incluye adicionalmente una base de datos esclava 260 a cargo de una tarea de proceso de servidor específico 220, 220', 220" presente en cada nodo esclavo 210, 210', 210" de control del desarrollo de nuevas versiones (Vn+1) 150 de los ficheros replicados que se almacenan inicialmente en el sistema de NAS 160 por cualquiera del servidor maestro 112 tras solicitud de un usuario administrativo del sistema de procesamiento de datos. La base de datos esclava tiene como objetivo esencialmente mantener el seguimiento de las versiones de los ficheros replicados mantenidos en la memoria compartida 230, 230', 230" de los nodos esclavos 210, 210', 210" de modo que se hace cumplir coherencia fuerte entre nuevas 150 versiones y actuales 250 de los ficheros replicados distribuidos en los nodos esclavos 210, 210', 210" por una parte y con los contenidos de la base de datos maestra 120 por otra parte. Ventajosamente, la base de datos esclava 260 únicamente almacena datos relacionados con la versión de ficheros replicados pero no almacena el contenido de los ficheros replicados.

El proceso que permite que se alcance este resultado sobre el sistema de procesamiento de datos ejemplar mostrado en la Figura 3 se explica en la Figura 4 posteriormente.

Todos los componentes de software que contribuyen a un sistema de acuerdo con la invención se implementan y ejecutan desde diversos ordenadores individuales o plataformas computarizadas 30, 30', 30", que también pueden comunicar a través de una o más redes interconectadas. Una red de este tipo es típicamente una red de área local o LAN que forma una red empresarial. La LAN se opera bajo un conjunto de normas de protocolos de comunicaciones. En la mayoría de casos se emplea el conjunto de protocolos de TCP/IP que asocia el protocolo de control de transmisión (TCP) de la Internet con el protocolo de Internet (IP) mismo. Las comunicaciones entre las diversas plataformas computarizadas y componentes de software se consigue en la práctica a través del intercambio de mensajes. Por ejemplo, los mensajes EDIFACT (intercambio de datos electrónicos para administración, comercio y transporte), una norma promovida por la organización de las Naciones Unidas (UN) y la asociación de transporte aéreo internacional (IATA), pueden usarse para establecer un canal de comunicación entre los diversos componentes del sistema de procesamiento de datos. Los mensajes se intercambian típicamente a través de un bus de servicio empresarial (no mostrado) también denominado como un integrador de servicio (SI).

La **Figura 4,** que está comprendida de las Figuras 4a y 4b, describe las etapas del método para conseguir consistencia fuerte entre los ficheros replicados distribuidos de un sistema de procesamiento de datos de acuerdo con la invención, un ejemplo de lo cual se ilustra y analiza en la figura anterior.

La primera fase del proceso de compromiso de 2 fases para actualizar datos en el sistema se muestra en la **Figura 4a.** Se activa por un usuario administrativo autorizado que inicia una transacción de actualización a través de uno de los servidores del nodo maestro 100. Esto se hace desde una interfaz de usuario convencional, en general una interfaz de usuario gráfica (GUI), bajo la forma del envío 1 de una solicitud de actualización 142, como se muestra en la **Figura 5,** al servidor maestro 112. Esto es la primera etapa o etapa 1 del proceso de actualización de datos.

La etapa 2 se realiza por el servidor maestro 112 que se encarga de, en la recepción de la solicitud de actualización 142, genera una nueva versión 150 (Vn+1) del correspondiente fichero replicado y lo almacena en el sistema de NAS 160. Para expedir la generación del nuevo fichero replicado 150, se realiza ventajosamente una actualización incremental de la versión anterior 250. Esto se explica adicionalmente en la Figura 5 posteriormente. Sea cual sea el método que se use, el resultado final de esta etapa es que se hace disponible una nueva versión (Vn+1) 150 del fichero replicado en el NAS 160 para todo el sistema de procesamiento de datos.

A continuación, en la etapa 3, debe realizarse una notificación de disponibilidad del nuevo fichero replicado 150. Esto

se hace a través de uno de los nodos esclavos 210, 210', 210", por ejemplo: el nodo esclavo 210 como se muestra en la Figura 3. Para este fin, se envía un mensaje por el servidor maestro 112 encargado a un proceso de servidor 220, 220', 220", es decir, una tarea de software específico que se ejecuta en este nodo y en todos los otros nodos esclavos 210', 210" también. A partir de ese momento, el nodo esclavo elegido 210 a continuación actúa como un punto de sincronización para todos los nodos esclavos 210, 210', 210" del sistema. Se denomina por lo tanto como el "nodo esclavo de sincronización" 210 en la siguiente descripción de la invención. Desde una perspectiva de nivel maestro todos los nodos esclavos desempeñan un papel idéntico. La elección de un nodo esclavo de sincronización está entonces basada típicamente en consideraciones de carga de trabajo.

La siguiente etapa, es decir, etapa 4, consiste en difundir a todos los otros nodos esclavos 210', 210", bajo el control del proceso de servidor 220 la tarea del nodo esclavo de sincronización 210, la notificación de disponibilidad de la nueva versión 150 del fichero replicado.

A continuación, en la etapa 5, tras la recepción de la notificación de disponibilidad del nuevo fichero replicado 150, cada nodo esclavo 210, 210', 210", incluyendo el nodo esclavo de sincronización 210, realiza las siguientes dos subetapas:

- Se hace una copia, es decir, una precarga 51 del fichero replicado nuevamente disponible 150 desde el sistema de NAS 160 en el que se ha almacenado previamente por el servidor maestro 112 a cargo de realizar la actualización.
- Una vez que se ha finalizado apropiadamente la precarga se envía un acuse de recibo 52 de vuelta al nodo esclavo de sincronización 210 por la tarea de proceso de servidor 220, 220', 220" de cada nodo esclavo 210, 210', 210".

La tarea de proceso de servidor 220 de sincronización del nodo esclavo 210 también está a cargo de recopilar todos los acuses de recibo de la finalización de precarga satisfactoria. Si, y cuando esto se consigue, se reenvía a su vez una notificación de consecución de precarga 6 satisfactoria en todos los nodos esclavos 210, 210', 210" al servidor maestro 112 a cargo. Esta es la etapa 6 del proceso de actualización global. En esta etapa, el nuevo fichero replicado 150 se ha precargado en memoria compartida 230, 230', 230" de cada nodo esclavo 210, 210', 210" y por lo tanto es potencialmente usable por la aplicación de clientes que se ejecuta en estos nodos. Sin embargo, se mantiene inactiva hasta que se realicen las siguientes etapas del proceso de actualización.

La segunda fase del proceso de actualización de datos en el sistema se muestra en la **Figura 4b.** La segunda fase se inicia tras la recepción por el servidor maestro, en la etapa 6, de una notificación de consecución de precarga satisfactoria del nuevo fichero replicado 150 por todos los nodos esclavos 210, 210', 210". Esta notificación de consecución de precarga se emite por la tarea de proceso de servidor 220 del nodo esclavo de sincronización 210. A continuación, el servidor maestro 112 a cargo almacena datos del nuevo fichero replicado 150 versión (Vn+1) en la base de datos maestra 120 en la etapa 7; por lo tanto, haciendo el uso de la nueva versión 150 para la totalidad del sistema de procesamiento de datos.

La etapa de compromiso necesita conseguirse adicionalmente, en la etapa 8, mediante el envío de un mensaje de compromiso emitido por el servidor maestro 112 hacia la tarea de proceso de servidor 220 del nodo esclavo de sincronización 210 para informar a este último de que el nuevo fichero replicado precargado (Vn+1) 150 está listo para usarse.

En la recepción del mensaje de compromiso, en la etapa 9, la nueva versión 150 del fichero de datos replicado de hecho también se hace en la base de datos esclava 260 por la tarea del proceso de servidor 220 del nodo de sincronización 210. Aún, en esta etapa, la nueva versión 150 ya precargada en cada nodo esclavo 210, 210', 210", no se usa realmente por las aplicaciones de cliente de nodos esclavos 210, 210', 210" hasta que tenga lugar la etapa de conmutación descrita posteriormente.

Sin embargo, en la etapa 10, el servidor maestro 112 recibe de la tarea de proceso de servidor 220 del nodo de sincronización 210 una respuesta para informarle de la finalización del compromiso en la base de datos esclava 260. Esta respuesta se propaga adicionalmente, en la etapa 11, al dispositivo de procesamiento de datos 140 del usuario administrativo del sistema de procesamiento de datos para informar a dicho usuario administrativo de que la actualización solicitada 142 de hecho se ha hacho y entra a partir de ahora en efecto.

El uso real de la nueva versión 150 del fichero replicado, sin embargo, tendrá efecto únicamente en el sistema cuando un proceso aplicativo realmente necesite responder a una consulta de información. En la presente invención, una consulta de información es una consulta que requiere recuperar datos almacenados en el sistema para que se cumpla mientras que una solicitud de actualización requiere actualizar los datos almacenados para que se cumpla. Por lo tanto una consulta de información se envía por el usuario final del sistema tal como un cliente o un agente de viajes mientras que la solicitud de actualización 142 se envía por usuarios administrativos a cargo de actualizar datos relacionados con productos, servicios o información a proporcionarse a usuarios finales. Tras la ejecución de un proceso aplicativo, la correspondiente aplicación de cliente siempre comprueba 264, desde la base de datos esclava 260, qué versión del fichero replicado necesario debe usarse. Si está disponible una nueva versión 150 a continuación se efectúa una conmutación a esta nueva versión 150, ya precargada en memoria compartida. La conmutación es una denominada

operación atómica en el sentido que cumple con este punto con las reglas ACID definidas para transacciones de base de datos, es decir: atomicidad, consistencia, aislamiento y durabilidad. Atomicidad significa que la conmutación se ejecuta de manera completa y no puede interrumpirse por cualquier otro proceso de modo que no se incurra en retardo por la aplicación de cliente. El mismo mecanismo se aplica en todos los procesos de aplicación. Esta solución garantiza consistencia completa de los datos usados desde cualquiera de las réplicas distribuidas. Como la conmutación es una operación atómica, no hay interrupción posible interrupción del proceso aplicativo mientras se recupera la nueva versión 150 de un fichero replicado. Puesto que la precarga del fichero replicado ha tenido lugar de manera temprana en la ejecución del proceso de actualización no hay retardo introducido cuando la aplicación cliente accede a la nueva versión 150 de un fichero replicado.

El flujo de proceso aplicativo puede modificarse ventajosamente para optimizar el número de solicitudes a la base de datos esclava 260. De hecho, es necesario hacer referencia a este punto de sincronización únicamente durante las transiciones, entre la precarga de una nueva versión de caché y el compromiso de esta nueva versión en la base de datos esclava 260.

Para este efecto, se añade un indicador de "precarga" a la memoria compartida durante la operación de precarga. Si este indicador no está presente, el proceso aplicativo accederá a la última versión disponible de caché, sin realizar una solicitud a la base de datos esclava 260. De otra manera, se realiza una solicitud a la base de datos esclava 260 y a continuación se usa la versión de la base de datos esclava 260.

Si la versión de precarga se ha hecho en la base de datos esclava 260, a continuación se realiza la operación de conmutación anteriormente descrita. En esta operación, se elimina el indicador de "precarga" de la memoria compartida.

El mecanismo descrito en figuras anteriores que explica cómo los ficheros replicados se hacen disponibles a los nodos esclavos 210, 210', 210" en tiempo casi real distingue entre dos tipos de comportamiento de los dos tipos de nodos del sistema:

- El nodo maestro 110 es el nodo escritor que está posiblemente comprendido de varios servidores maestros 112. Cada uno de ellos se hace apto para gestionar transacciones emitidas a través de la interfaz de usuario 140 por cualesquiera usuarios administrativos del sistema de procesamiento de datos. Las transacciones de gestión se pretenden para actualizar eventualmente la base de datos maestra 120. Para este fin, como ya se ha analizado, una transacción de gestión crea en primer lugar una nueva versión de fichero replicado (Vn+1) 150 y lo almacena en el sistema de NAS 160. La invención supone que el nodo maestro 110 no tiene conocimiento de la topología de los nodos esclavos 210, 210', 210" y no es responsable de extraer datos más allá en los nodos esclavos 210, 210', 210". Por lo tanto, la escalabilidad de la aplicación 200 es totalmente independiente del nodo maestro 110.
- En su lugar, los nodos esclavos 210, 210', 210" consiguen por ellos mismos una copia de sólo lectura de los ficheros replicados precargándolos directamente desde el sistema NAS compartido 160. La información de difusión a los nodos esclavos 210, 210', 210" se hace a través del bus de servicio empresarial o integrador de servicio (SI) mencionado previamente. Para mantener el rendimiento tan alto como sea posible el volumen de datos difundidos se mantiene ventajosamente a un mínimo. Para conseguir este objetivo, el método de la invención gestiona únicamente notificaciones de difusión de disponibilidad de nuevos ficheros replicados a los nodos esclavos 210, 210', 210" permitiéndoles por lo tanto copiar el nuevo fichero replicado 150 de una estructura de fichero compartido común, el NAS 160, como se explica en la Figura 4.

La replicación en tiempo casi real implica que todos los nodos clientes pueden de hecho conmutar a una nueva versión 150 del fichero replicado tan pronto como finaliza la transacción de gestión, es decir, en la etapa 11 como se analiza en la Figura 4. Este tipo de replicación también es conocida como ser un tipo de replicación activa que está bastante más restringida que otros tipos de replicación menos exigentes a menudo calificadas como replicación perezosa. Para obtener este resultado, la invención separa el flujo de datos en un flujo de control y un flujo de datos. Por lo tanto, como ya se ha indicado anteriormente, las notificaciones de versiones de fichero replicado para su uso se difunden a los nodos cliente mediante el integrador de servicio. Los datos concernientes de ficheros replicados se hacen disponibles a partir del sistema de ficheros compartido, es decir, el sistema de NAS 160. Únicamente el flujo de control que lleva versiones necesita transportarse y encaminarse a través de los nodos esclavos de sincronización 210 a todos los otros nodos esclavos 210', 210" del nivel de aplicación 200, mientras que los datos pueden transferirse directamente en paralelo a cada nodo esclavo 210, 210', 210". Esta arquitectura cumple el requisito de crear una separación entre servidores 112 del nodo maestro 100, es decir, los escritores, y los nodos esclavos 210, 210', 210". Por lo tanto, los escritores necesitan únicamente almacenar los nuevos datos, primero en el sistema de ficheros compartido de NAS, a continuación en la misma base de datos maestra 120. Los escritores, no se requiere que tengan ningún conocimiento de la topología del nodo esclavo 210, 210', 210". Los servidores maestros, es decir, escritores, se basan en el integrador de servicio para encaminar información de versionado de ruta a los nodos esclavos 210, 210', 210".

Escribir en la estructura de fichero compartido de NAS se ilustra en la **Figura 5.** Describe la actualización incremental que se realiza en la etapa 2 cuando se crea una nueva versión (Vn+1) 150 de un fichero replicado.

Haciendo disponibles los ficheros replicados, es decir, publicándolos a partir de una estructura de fichero compartida tal como un sistema de NAS 160 proporciona beneficios adicionales. El servidor maestro 112 a cargo de manejar la solicitud de actualización 142 puede acceder fácilmente a la estructura de fichero compartido de NAS para obtener 152 la última versión 250 de un fichero replicado actualmente usado por todos los nodos esclavos 210, 210', 210". La estructura binaria de NAS 160 está ideada de manera ventajosa para facilitar la recuperación de ficheros replicados y también para minimizar requisitos de almacenamiento. Por lo tanto, los ficheros replicados están posiblemente comprimidos para limitar adicionalmente la cantidad global de almacenamiento necesaria para mantener todos ellos. Puede tener que ejercerse una compensación entre tiempo de acceso y requisitos de almacenamiento que permite la recuperación de ficheros replicados dentro de un tiempo de acceso compatible con los rendimientos esperados del sistema de procesamiento de datos mientras se limita de alguna manera la cantidad global de almacenamiento requerido. El consumo de ancho de banda de red para transferir los ficheros también se tiene en cuenta. Sea cual sea el compromiso que se adopte, los ficheros replicados recuperados necesitan convertirse por el servidor maestro al modelo de objeto de negocio (BOM) 114 en el que pueden funcionar en la memoria de servidor. El servidor maestro 112 puede a continuación realizar una actualización incremental 116 añadiendo la actualización solicitada a la versión actual de un fichero replicado para incrustar las modificaciones solicitadas por un usuario administrativo en la solicitud de actualización 142.

Más precisamente el fichero replicado recuperado necesita convertirse por el servidor maestro a un Modelo de Objeto de Negocio de C++ 114 en la memoria de servidor. El servidor maestro 112 puede a continuación realizar una actualización incremental en esta estructura C++ 116 añadiendo la actualización. A continuación se crea un nuevo fichero replicado basándose en esta nueva estructura C++.

En comparación con la recuperación de la información correspondiente de la base de datos maestra 120, esta manera de hacer acelera considerablemente el proceso de generar nuevas versiones de ficheros replicados. Puede conseguirse una mejora promedio de un orden de magnitud (x10). El fichero replicado actualizado finalmente se convierte y almacena de vuelta 154 en la estructura de fichero compartida de NAS 160.

Lo siguiente analiza adicionalmente lo que significa consistencia fuerte en el contexto de la invención y cómo se obtiene con el sistema y el método anteriormente descritos sobre todos los ficheros replicados de nodos esclavos 210, 210', 210" cada uno implementado en una máquina física diferente 30, 30', 30".

En un sistema de procesamiento de datos de la invención la base de datos maestra 120 se supone que se actualiza de manera regular. Aunque esta figura puede variar ampliamente dependiendo de la aplicación de la invención, una frecuencia típica de actualizaciones a considerar está en el orden de un par de veces por minuto. Cada actualización produce una nueva versión 150 de un fichero replicado (Vn+1). Entonces, se consigue consistencia fuerte si, a partir del momento de un fichero replicado de una versión dada, es decir, la versión N, se lee en un nodo esclavo dado, cualquier lectura posterior en cualquier otro nodo esclavo 210', 210" de hecho devuelve la misma versión de fichero replicado N. De manera evidente, una versión superior podría leerse también si, mientras tanto, ya se ha distribuido una nueva versión 150.

La consistencia se consigue proporcionando a todos los nodos esclavos 210, 210', 210" un punto de acceso central que mantiene el registro de las versiones activas de todos los ficheros replicados. Este punto de acceso se implementa preferentemente como una tabla de una base de datos operada bajo un sistema de gestión de base de datos relacional convencional (RDBMS) para garantizar que se cumplan las propiedades ACID. En un sistema de la invención, como se muestra en la Figura 3, este papel se desempeña por la base de datos esclava 260. Se muestra un extracto de la tabla de versión almacenada 262. En el ejemplo ilustrado, BA-NGI corresponde a Vn+1 que significa que debe usarse la versión Vn+1 del fichero replicado, es decir, datos de la aplicación NGI operada por el proveedor de viaje British Airways (BA). NGI, acrónimo para Inventario de la Nueva Generación, discrimina qué aplicación particular es la propietaria, es decir, responsable, para los datos de regla dada. La tabla de versión central almacenada en la base de datos esclava se actualiza con el proceso de compromiso de 2 fases descrito en la Figura 4.

A continuación, en el inicio de cada transacción indicada por una aplicación cliente que se ejecuta en un nodo esclavo, la versión del fichero replicado activo se captura 264 de la tabla de versión 262 de la base de datos esclava 260. Si la versión de fichero replicado activo hallada es mayor que la actualmente usada, el nodo esclavo 210, 210', 210" conmuta inmediatamente al nuevo fichero replicado (Vn+1) 150. Como se explica en la Figura 4, esto es posible puesto que el fichero replicado actualizado 150 ya se ha precargado en memoria compartida 230, 230', 230" del nodo esclavo 210, 210', 210". Vale la pena señalar aquí que si la transacción iniciada por una aplicación cliente requiere varios accesos a un fichero replicado, el sistema gestiona para usar la misma versión a través de todo el tiempo de vida de esta transacción incluso aunque se haga disponible una nueva versión 150 mientras tanto. Esto cumple la regla de aislamiento del conjunto de reglas ACID que se aplican a transacciones de base de datos.

Para proporcionar un nivel incluso superior de rendimiento al acceso de datos en memoria compartida puede omitirse opcionalmente la interrogación anterior 264 de la base de datos esclava en el comienzo de cada transacción. En este caso, los ficheros replicados almacenados en memoria compartida necesitan contener una etiqueta o bandera que indica si está disponible o no una nueva versión 150. Por lo tanto, es esta etiqueta la que se comprueba en primer lugar en el comienzo de cada transacción en lugar de interrogar sistemáticamente la base de datos esclava. Si la

etiqueta indica que no está disponible la nueva versión 150, entonces se usa el fichero replicado actual. Si la etiqueta indica que una nueva versión 150 está de hecho disponible, a continuación se lee la tabla de versión 262 de la base de datos esclava de modo que el proceso aplicativo 240, 240', 240" puede conmutar al nuevo fichero replicado precargado (Vn+1). La etiqueta, que se estableció cuando se hizo la precarga de la nueva versión 150, se resetea cuando la aplicación de cliente ha finalizado de procesar su transacción actual. En un entorno donde numerosos procesos están accediendo al mismo fichero o ficheros replicados, únicamente unos pocos procesos tendrán que consultar la base de datos antes de que uno de ellos resetee la etiqueta.

Para funcionar en un entorno de producción el mecanismo anterior descrito tiene que ser también tolerante a fallos. La consistencia completa puede alcanzarse únicamente si los ficheros replicados se han precargado realmente todos satisfactoriamente, es decir, distribuidos y mapeados en memorias compartidas de todos los nodos esclavos antes de que pueda hacerse la transacción de gestión en la base de datos maestra. Esto es obligatorio para asegurar que esos nodos esclavos pueden de hecho conmutar al nuevo fichero replicado de una manera atómica. De otra manera, puede haber casos de error donde algunos esclavos podrían conmutar a la nueva versión y otros no, produciendo de esta manera una ausencia de consistencia. Por lo tanto, para evitar que ocurra esta situación, si por cualquier razón la distribución falla en cualquier nodo esclavo dado, se aborta la transacción total y se envía un error al usuario final en respuesta a la solicitud de actualización. Esto es posible debido al proceso de compromiso de 2 fases puesto en su lugar y descrito en la Figura 4. Entre consistencia y disponibilidad la invención pone énfasis en la consistencia. Es importante destacar que leer la disponibilidad de nodos esclavos, sin embargo, aún está completamente garantizada y puede satisfacerse consulta de información casi en tiempo real. La única parte de disponibilidad que posiblemente está comprometida es la actualización/gestión de datos. En el contexto de la invención esto es bastante menos crítico que disponibilidad de lectura y consistencia fuerte de ficheros replicados distribuidos a través de todos los nodos esclavos. La distribución de ficheros replicados satisfactoria a cada nodo esclavo es por lo tanto una parte crítica de la invención. El bus de servicio empresarial o integrador de servicio (SI) previamente analizado se usa para asegurar que una imagen del grupo de nodos esclavos que participan de manera activa en el nivel de aplicación de cliente está siempre disponible. Cuando un nodo esclavo se une o deja el grupo, la imagen se actualiza en consecuencia de modo que el método de la invención puede llevarse a cabo de manera apropiada.

Mientras que se ha mostrado y descrito la presente realización preferida de la invención, debe entenderse claramente que esta invención no se limita a esto, sino que puede ponerse en práctica de manera diversa dentro del alcance de las siguientes reivindicaciones.

REIVINDICACIONES

1. Un método de mantenimiento de la consistencia de ficheros replicados (250) distribuidos a través de una pluralidad de nodos esclavos de procesamiento independientes (210, 210', 210") que es una parte de un nivel de aplicación (200) de un sistema de procesamiento de datos de cliente/servidor de múltiples clientes, estando distribuidos los ficheros replicados (250, 150) desde un nodo maestro (110) de un nivel maestro (100), el método **caracterizado por que** comprende las siguientes etapas realizadas con al menos un procesador de datos:

recibir (1) una solicitud de actualización (142) en al menos un servidor maestro (112) del nivel maestro (100) para actualizar una base de datos maestra (120) del sistema de procesamiento de datos;
basándose en dicha solicitud de actualización generar y almacenar (2) una nueva versión (150) de un fichero replicado almacenado en un sistema de ficheros compartido (160) del nivel maestro, comprendiendo los ficheros replicados contenidos de la base de datos maestra (100);
proporcionar (3, 4) una notificación de disponibilidad de la nueva versión (150) del fichero replicado a todos los nodos esclavos (210, 210', 210"); en el que la etapa de proporcionar (3, 4) una notificación de disponibilidad de la nueva versión (150) del fichero replicado a todos los nodos esclavos (210, 210', 210") comprende las siguientes etapas:

enviar (3), desde el servidor maestro (112), una notificación de disponibilidad de la nueva versión (150) del fichero replicado a un nodo esclavo de sincronización (210) tomada entre los nodos esclavos (210, 210', 210");
difundir (4) desde el nodo esclavo de sincronización (210) dicha notificación de disponibilidad a todos los otros nodos esclavos (210', 210");

en cada nodo esclavo (210, 210', 210"), iniciar la precarga (51) desde el sistema de ficheros compartido (160) de la nueva versión (150) del fichero replicado y tras la finalización de la precarga, realizar acuse de recibo (52) al nodo esclavo de sincronización de la finalización satisfactoria de la precarga;
únicamente si todos los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga realizar a continuación de manera sucesiva las siguientes etapas:

recibir en el servidor maestro (112) una notificación de consecución de precarga (6) desde el nodo esclavo de sincronización;
desde el servidor maestro (112), actualizar (7) la base de datos maestra (120) con datos de la nueva versión (150) del fichero replicado haciéndose por lo tanto el uso de la nueva versión (150) del fichero replicado por el nivel maestro (100); desde el servidor maestro que envía un mensaje de compromiso (8) al nodo de sincronización;

en el nodo esclavo de sincronización (210), hacer (9) el uso de la nueva versión (150) del fichero replicado en una base de datos esclava (260) del nivel de aplicación (200) dispuesto para mantener el registro de todas las versiones de fichero replicado (150, 250);
desde el nodo esclavo de sincronización (210), responder al servidor maestro (112) una notificación de compromiso del uso de la nueva versión (150) del fichero replicado en la base de datos esclava (10); desde el servidor maestro (112), que realiza acuse de recibo (11) de la consecución de la actualización en respuesta a la solicitud de actualización recibida (142),
si no todos los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga, recibir a continuación en el nodo maestro (110) una notificación de error.

2. El método de acuerdo con la reivindicación anterior en el que la etapa de realizar acuse de recibo (52) de la finalización satisfactoria de la precarga comprende una etapa en la que los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga a un proceso de servidor (220) del nodo esclavo de sincronización (210).

3. El método de acuerdo con la reivindicación anterior en el que una vez que todos los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga al proceso de servidor (220) del nodo esclavo de sincronización (210) y antes de la etapa de recepción en el servidor maestro (112) de una notificación de consecución de precarga (6), el nodo esclavo de sincronización (210) reenvía al servidor maestro (112) dicha notificación de consecución de precarga (6).

4. El método de acuerdo con una cualquiera de las tres reivindicaciones anteriores en el que tras la finalización de la etapa de hacer el uso de la nueva versión (150) del fichero replicado por el nivel maestro (100), el nodo maestro (110) reenvía una notificación de compromiso (8) al nodo esclavo de sincronización (210) y a continuación el nodo esclavo de sincronización (210) activa la etapa de hacer (9) el uso de la nueva versión (150) del fichero replicado en la base de datos esclava (260).

5. El método de acuerdo con una cualquiera de las reivindicaciones anteriores en el que en el comienzo de una transacción iniciada por una consulta para información de un usuario, al menos en cada nodo esclavo (210, 210', 210") implicado en dicha transacción de un proceso aplicativo (240, 240', 240") interroga (264) la base de datos esclava

(260) y se da instrucción de si conmutar (12) desde una versión actual (250) del fichero replicado a la nueva versión precargada (150) del fichero replicado.

6. El método de acuerdo con una cualquiera de las reivindicaciones 1 a 4, en el que en el comienzo de una transacción iniciada por una consulta para información desde un usuario, en cada nodo esclavo (210, 210', 210") implicado en dicha transacción de un proceso aplicativo (240, 240', 240") lee una etiqueta de una versión actual (250) del fichero replicado, indicando dicha etiqueta si una nueva versión (250) del fichero replicado está disponible, y en el que si la etiqueta indica que una nueva versión (150) del fichero replicado está disponible, a continuación el proceso aplicativo (240, 240', 240") interroga (264) la base de datos esclava (260) y se da instrucción de si conmutar (12) de la versión actual (250) del fichero replicado a la nueva versión (150) del fichero replicado.

7. El método de acuerdo con la reivindicación anterior en el que la etiqueta se establece en la finalización satisfactoria de precarga (51) de una nueva versión (150) del fichero replicado y se resetea en la primera lectura por el proceso aplicativo (240, 240', 240").

8. El método de acuerdo con una cualquiera de las reivindicaciones anteriores en el que la solicitud de actualización recibida (142) comprende al menos una actualización y en el que las etapas de generar y almacenar una nueva versión (150) de un fichero replicado en el sistema de ficheros compartido (160) son un proceso incremental que incluye, en el servidor maestro (112), las etapas de:

recuperar del sistema de ficheros compartido (160) una versión actual (250) de un fichero replicado y convertir (152) a un formato apropiado (114) la versión actual del fichero replicado para que se actualice;
aplicar la actualización (142) en la versión actual y convertir y almacenar (154) la nueva versión (150) del fichero replicado en el sistema de ficheros compartido (160).

9. El método de acuerdo con la reivindicación anterior en el que los ficheros replicados (150, 250) se comprimen cuando se almacenan en el sistema de ficheros compartido (160) y en el que los ficheros replicados almacenados en el sistema de ficheros compartido (160) se descomprimen y convierten en un formato apropiado por el servidor maestro (112) cuando se realiza la actualización incremental (152), y por los nodos esclavos (210, 210', 210") tras la precarga (51).

10. El método de acuerdo con la reivindicación 1, en el que si no todos los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga, a continuación el nodo maestro (110) envía una notificación de error a un administrador que envió dicha solicitud de actualización (142).

11. El método de acuerdo con una cualquiera de las reivindicaciones anteriores en el que el sistema de ficheros compartido (160) es un almacenamiento de conexión en red (NAS).

12. El método de acuerdo con una cualquiera de las reivindicaciones anteriores en el que la base de datos esclava (260) únicamente almacena datos relacionados con la versión de ficheros replicados.

13. Producto de programa informático almacenado en un medio de memoria legible por ordenador y que realiza el método de acuerdo con una cualquiera de las reivindicaciones anteriores.

14. Un sistema de procesamiento de datos de cliente/servidor de múltiples niveles distribuido que comprende un nivel de aplicación (200) y un nivel maestro (100), comprendiendo el nivel de aplicación (200) una pluralidad de nodos esclavos de procesamiento independientes (210, 210', 210") y comprendiendo el nivel maestro (100) un nodo maestro (110) dispuesto para distribuir ficheros replicados (250, 150) a los nodos esclavos (210, 210', 210"), una base de datos maestra (120) y un servidor maestro (112), **caracterizado por que:**

el nivel maestro (100) comprende un sistema de ficheros compartido (160), estando dispuesto el servidor maestro (112) para recibir (1) una solicitud de actualización (142) para actualizar la base de datos maestra (120), para generar una nueva versión (150) de un fichero replicado almacenado en el sistema de ficheros compartido (160) y para almacenar (2) dicha nueva versión (150) en el sistema de ficheros compartido (160); **por que** comprende una base de datos esclava (260) conectada a todos los nodos esclavos (210, 210', 210") y dispuesta para mantener registro de todas las versiones de fichero replicado (150, 250); y **por que**
el sistema está configurado para:
proporcionar (3, 4) una notificación de disponibilidad de la nueva versión (150) del fichero replicado a todos los nodos esclavos (210, 210', 210"); estando configurado el sistema de modo que la etapa de proporcionar (3, 4) una notificación de disponibilidad de la nueva versión (150) del fichero replicado a todos los nodos esclavos (210, 210', 210") comprende las siguientes etapas:

enviar (3), desde el servidor maestro (112), una notificación de disponibilidad de la nueva versión (150) del fichero replicado a un nodo esclavo de sincronización (210) tomada entre los nodos esclavos (210, 210', 210");
difundir (4) desde el nodo esclavo de sincronización (210) dicha notificación de disponibilidad a todos los otros nodos esclavos (210', 210");

en cada nodo esclavo (210, 210', 210"), iniciar la precarga (51) desde el sistema de ficheros compartido (160) de la nueva versión (150) del fichero replicado y tras la finalización de la precarga, realizar acuse de recibo (52) de la finalización satisfactoria de la precarga al nodo esclavo de sincronización;

únicamente si todos los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga realizar a continuación de manera sucesiva las siguientes etapas: recibir en el servidor maestro (112) una notificación de consecución de precarga (6) del nodo esclavo de sincronización; desde el servidor maestro (112), actualizar (7) la base de datos maestra (120) con datos de la nueva versión (150) del fichero replicado haciendo por lo tanto el uso de la nueva versión (150) del fichero replicado por el nivel maestro (100); desde el servidor maestro que envía un mensaje de compromiso (8) al nodo de sincronización;

en el nodo esclavo de sincronización (210), hacer (9) el uso de la nueva versión (150) del fichero replicado en la base de datos esclava (260); desde el nodo esclavo de sincronización (210), responder al servidor maestro (112) una notificación de compromiso del uso de la nueva versión (150) del fichero replicado en la base de datos esclava (10); desde el servidor maestro (112), realizar acuse de recibo (11) de la consecución de la actualización en respuesta a la solicitud de actualización recibida (142),

si no todos los nodos esclavos (210, 210', 210") realizan acuse de recibo (52) de la finalización satisfactoria de la precarga, recibir a continuación en el nodo maestro (110) una notificación de error.

Técnica Anterior

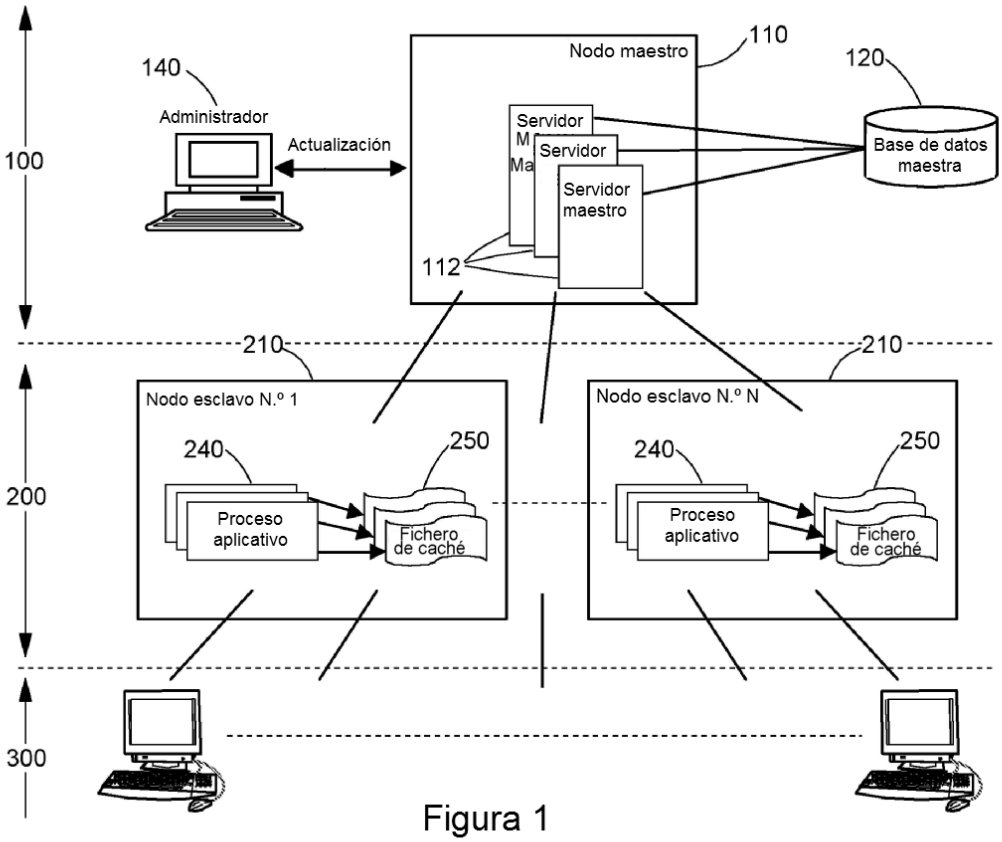

Figura 1

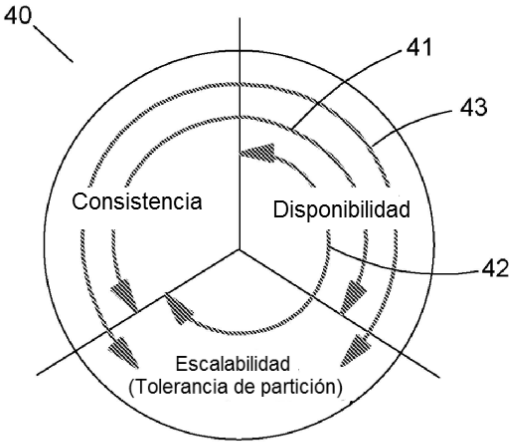

Figura 2

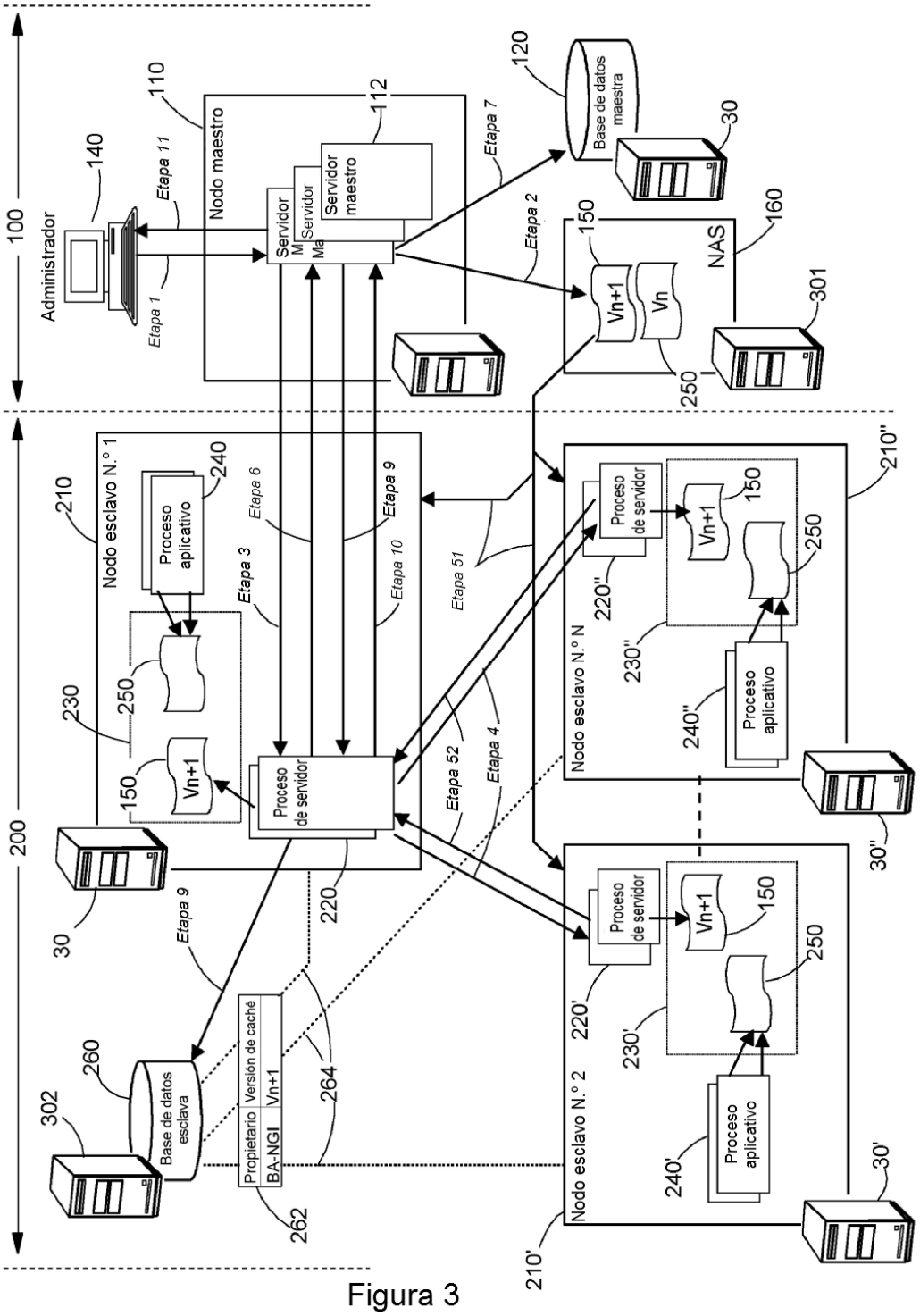
100
200
Administrador
140
Etapa 1
Etapa 11
Nodo maestro
110
Servidor M
Servidor Ma
Servidor maestro
112
Etapa 7
120
Base de datos maestra
30
Etapa 2
150
Vn+1
Vn
NAS
160
301
250
Nodo esclavo N.º 1
210
Proceso aplicativo
240
Etapa 6
Etapa 3
Etapa 9
Etapa 10
Etapa 51
230
250
Vn+1
150
Proceso de servidor
220
Etapa 52
Etapa 4
30
Etapa 9
260
Base de datos esclava
302
Propietario
Versión de caché
BA-NGI
Vn+1
264
262
Nodo esclavo N.º N
210''
Proceso de servidor
220''
Vn+1
150
250
230''
240''
Proceso aplicativo
30''
Nodo esclavo N.º 2
210'
220'
230'
240'
30'

Figura 3

1ª fase de proceso de compromiso de 2 fases:

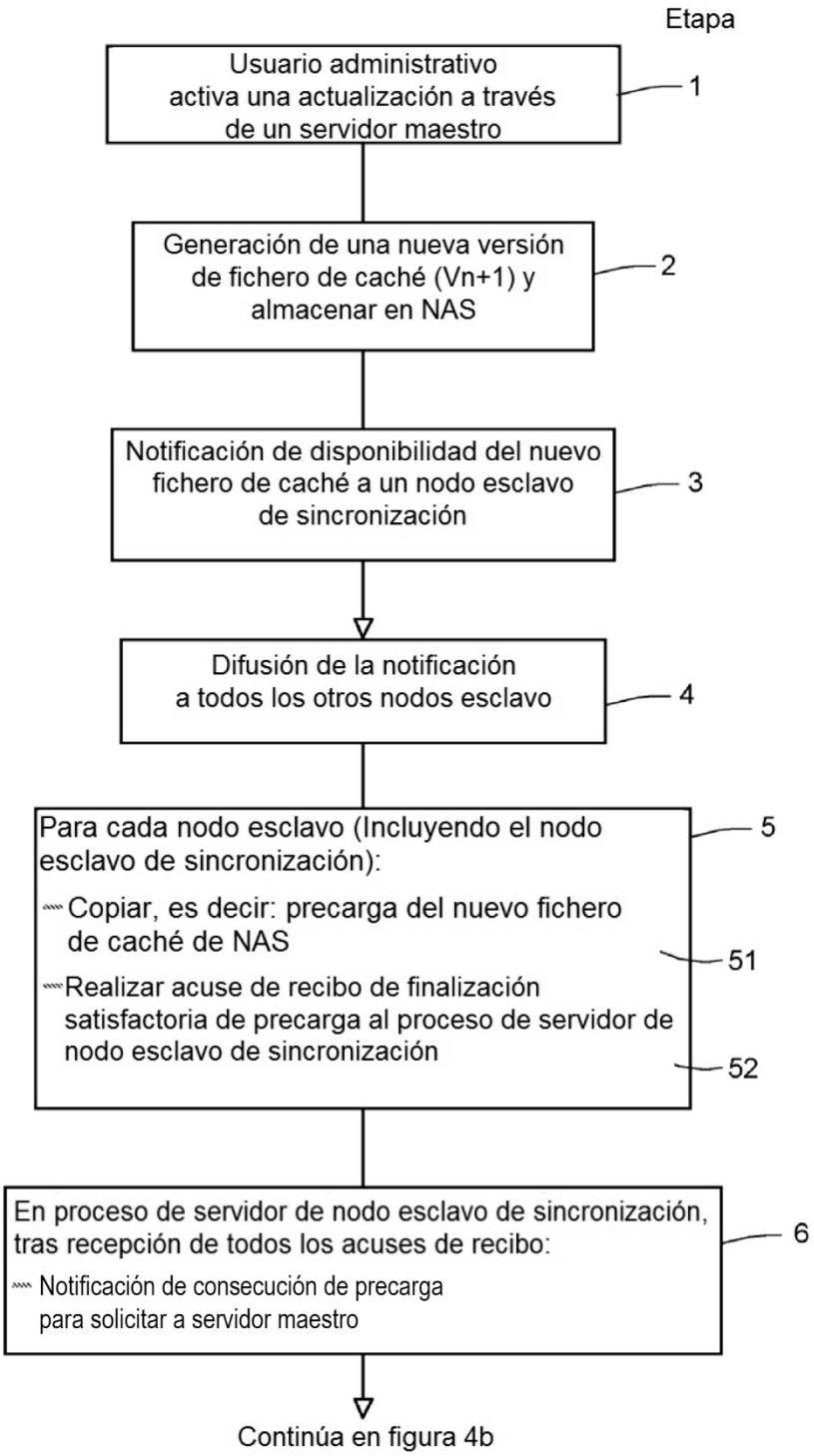

Figura 4a

2ª fase de proceso de compromiso de 2 fases:

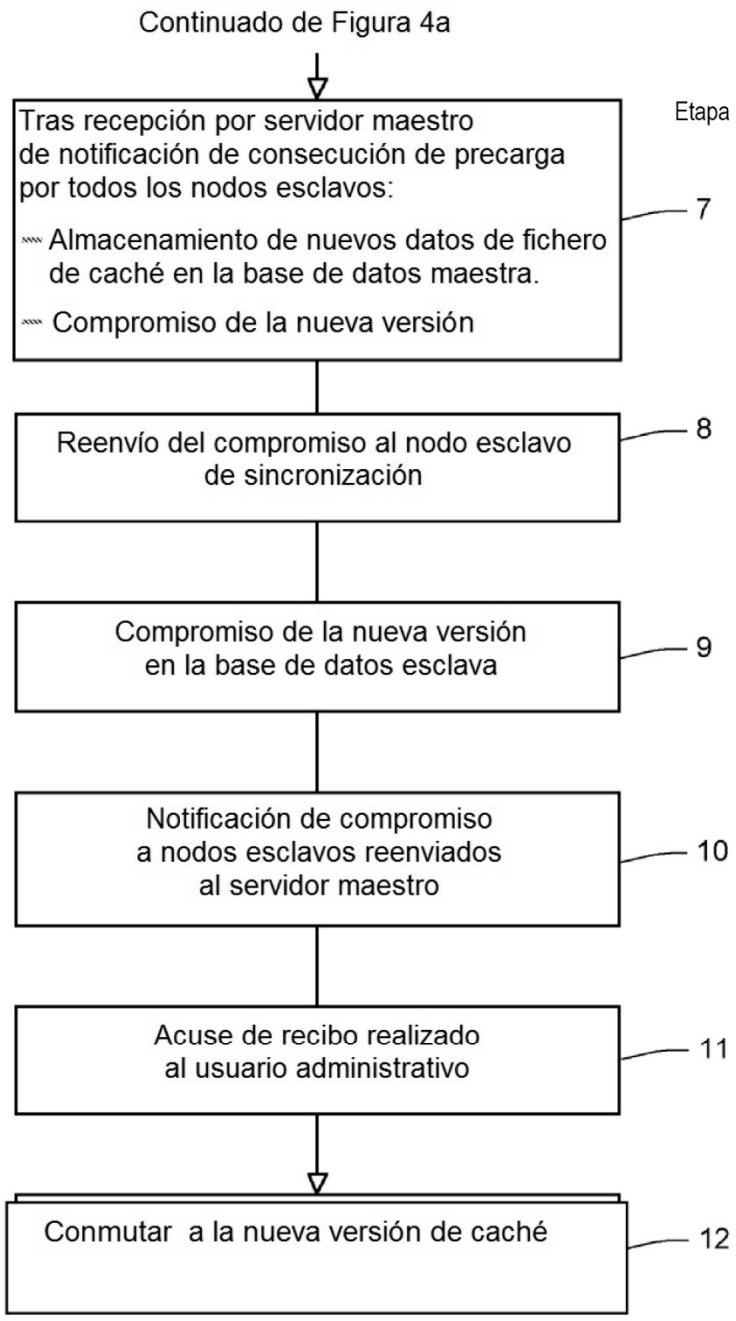

Figura 4b

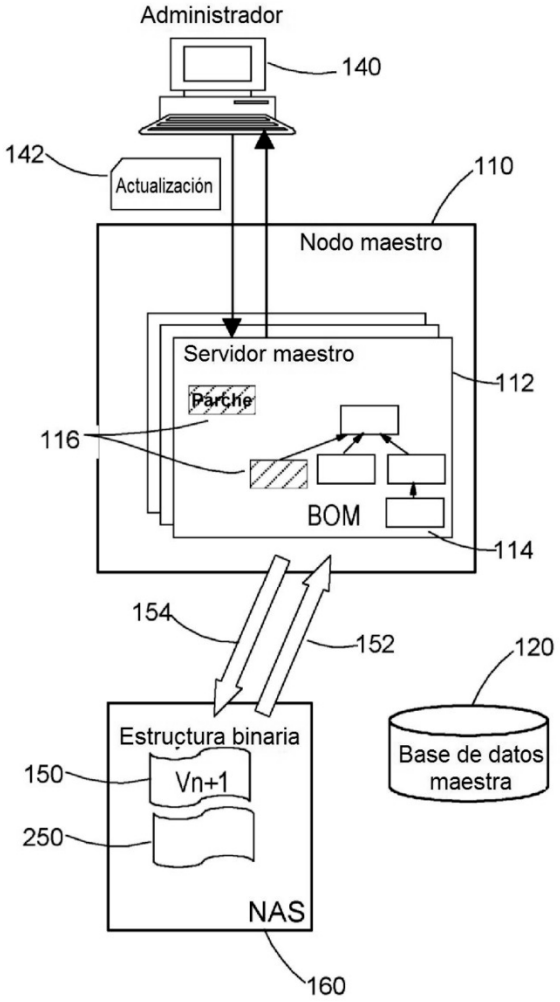
Administrador
140
142
Actualización
110
Nodo maestro
Servidor maestro
112
Parche
116
BOM
114
154
152
120
Estructura binaria
Base de datos maestra
150
Vn+1
250
NAS
160

Figura 5