(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 696 604**

(51) Int. CI.:

G06F 11/14 (2006.01)
G06F 11/18 (2006.01)
G06F 11/07 (2006.01)
G06F 11/16 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **13.12.2013 E 13290313 (9)**

(97) Fecha y número de publicación de la concesión europea: **15.08.2018 EP 2884392**

(54) Título: **Arquitectura de marco tolerante a fallos con triple redundancia de software**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
17.01.2019

(73) Titular/es:

THALES (100.0%)
Tour Carpe Diem - Place des Corolles, Esplanade Nord
92400 Courbevoie, FR

(72) Inventor/es:

DE OLIVEIRA, JAIME;
ESTAVES, GUY;
TOURTEAU, FABIAN y
SCHERRER, CHRISTOPH

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Arquitectura de marco tolerante a fallos con triple redundancia de software

Campo técnico

Esta patente se refiere al campo del procesamiento digital de datos y más específicamente al campo de los sistemas de tolerancia a fallos.

Antecedentes de la técnica

El funcionamiento de un procesador en el espacio, por ejemplo, en un satélite, es crucial para el cumplimiento de la misión y para la integridad de los datos. Las limitaciones ambientales conducen a tratar de mejorar el control de la tolerancia de los sistemas a los fallos.

En la actualidad, el problema técnico de sensibilidad a la radiación se mitiga con soluciones basadas en las arquitecturas de redundancia modular triple (TMR) y/o los circuitos integrados digitales “rad-hard” (es decir, radiaciones endurecidas) específicos. Las arquitecturas TMR son técnicas bien conocidas de técnicas de tolerancia a fallos aplicables al diseño de circuitos integrados digitales de la arquitectura de nivel de sistema que consiste en replicar tres veces un bloque informático físico que realiza la misma tarea y votar sus salidas. Un circuito integrado digital rad-hard es un enfoque (nivel de fabricación) que consiste en usar tecnologías de procedimiento específicas (por ejemplo, silicio sobre aislante - SOI) o/y patrones de diseño de circuitos para mejorar la tolerancia a fallos del procesador. Ambos enfoques proporcionan una eficacia comprobada de tolerancia a fallos por radiación, pero presentan varias desventajas.

Estos enfoques son extremadamente caros (por ejemplo, en términos de costes de adquisición y costes de ingeniería corriente abajo). Se basan en tecnologías antiguas que ofrecen muy poca potencia de procesamiento y no aprovechan las tecnologías comerciales de alto rendimiento.

A menudo dependen también de la misión, es decir, no están diseñados para la escalabilidad y por lo tanto no pueden reutilizarse.
El documento de patente publicado EP2498184 desvela un dispositivo que tiene una capa de software, es decir, un hipervisor (202), que centraliza los intercambios entre un procesador y una aplicación (201) y que implementa mecanismos de gestión de tolerancias a fallos. Un componente electrónico programable forma una interfaz entre el procesador y una unidad de memoria, por ejemplo, una RAM dinámica síncrona y una interfaz de entrada y salida de datos. Uno de los mecanismos es una función de restablecimiento en el estado conocido del procesador, donde la función es periódica con un período configurable. El mecanismo se restablece al estado conocido mediante una señal de reinicio emitida por el componente electrónico programable. Este enfoque presenta limitaciones. La publicación de RICARDO PAHARSINGH Y COL.: “An Availability Model of a Virtual TMR System with Applications in Cloud/Cluster Computing”, INGENIERÍA DE SISTEMAS DE ALTA SEGURIDAD (HASE), 2011 IEEE 13º SIMPOSIO INTERNACIONAL, IEEE, 10 de noviembre de 2011 (10-11-2011), páginas 261-268, XP032083396, desvela un modelo de un “Virtual TRM System with Applications in Cloud/ Cluster Computing”, que comprende la ejecución de software equivalente en máquinas virtuales. El documento de patente US20080148015 desvela un sistema que incluye una pluralidad de procesadores multinúcleo, una tabla para gestionar los procesadores y los núcleos incluidos en los procesadores [que] se proporcionan y un único servidor virtual [que] se forma usando los núcleos incluidos en diferentes procesadores al generar el servidor virtual. De acuerdo con el número incluido en los procesadores, el número de procesadores varía, si se detecta un preeco de fallo en un procesador, se ejerce un control con el fin de no entregar una programación de ejecución de un mecanismo de virtualización al procesador en el que se ha detectado el preeco de fallo.

Por lo tanto, existe la necesidad de soluciones para resolver los problemas mencionados anteriormente. Las realizaciones de la presente invención ofrecen tales soluciones, al menos en parte.

Sumario de la invención

La invención se define por las reivindicaciones independientes adjuntas.

Se desvela un procedimiento implementado por ordenador para detectar un fallo en un sistema que comprende las etapas de ejecutar al menos tres máquinas virtuales, ejecutando cada máquina virtual un mismo software de aplicación, en segmentos de memoria separados y aislados y en un núcleo dedicado de un procesador multinúcleo; estando dichas máquinas virtuales sincronizadas y ejecutándose simultáneamente por un hipervisor común; en el que las máquinas virtuales no defectuosas proporcionan un mensaje de salida idéntico dentro de un intervalo de tiempo predefinido; detectar un fallo en una salida de una máquina virtual, correspondiendo dicha fallo a un mensaje de salida diferente de dicha máquina virtual defectuosa. Los desarrollos incluyen un mecanismo de voto distribuido, mecanismos de insertar/extraer, asociación de mensajes de voto de salida con una extensión de seguridad que comprende una información de identificación y recuperación de máquina virtual usando contexto de datos.

Ejemplos proporcionados de este modo desvelan una arquitectura de marco de software basándose en tecnologías

de virtualización integradas que permiten una arquitectura SW/HW tolerante a fallos combinada basada en procesadores multinúcleo COTS. Las ventajas asociadas con las realizaciones de la invención son numerosas.

Pueden usarse procesadores (componente sacado del estante - COTS) comerciales baratos y más rápidos. Por ejemplo, las aplicaciones de carga útil de a bordo de satélite pueden usar procesadores modernos, en lugar de los tradicionales circuitos integrados digitales “rad-hard”. En particular, pueden usarse procesadores multinúcleo (tales procesadores no se fabrican para tales condiciones de radiación existentes en el dominio del espacio). Algunas realizaciones permiten por lo tanto habilitar el procesamiento de datos de gran ancho de banda y el cálculo intensivo de algoritmos. Se mejora la tolerancia de estos procesadores multinúcleo a los fallos inducidos por radiación en el dominio del espacio.

Al mismo tiempo, se conserva la tolerancia de software de aplicación de carga útil (software de aplicación (ASW)) a los fallos inducidos por radiación (es decir, preservar la tolerancia a los fallos inducidos por radiación). En general, la integridad de los datos de salida está garantizada. La plataforma informática basada en COTS de acuerdo con las realizaciones de la invención alcanza niveles comparables de disponibilidad (y confiabilidad) como con las soluciones de hardware rad-hard redundantes.

La ingeniería de a bordo de satélites se reduce, así como los costes de compra o adquisición mientras que se preserva la eficacia de tolerancia a fallos de carga útil de aplicación para los fallos inducidos por la radiación.

Las realizaciones de la invención, en general, proporcionan sistemas flexibles, escalables y reutilizables en misiones críticas de dominio del espacio.

La arquitectura SW/HW mixta descrita es independiente de la misión, es decir, genérica y por lo tanto reutilizable. La compensación de “tamaño, peso y potencia” (SWaP) está optimizada. Se mejora la relación de consumo rendimiento/potencia y la reducción del tamaño y el peso del hardware de a bordo. Las tarjetas informáticas pueden comprender menos (y mejores) chips que participen para reducir el tamaño y el peso de los satélites.

De acuerdo con una primera realización de la presente invención, se proporciona un procedimiento como se describe adicionalmente en la reivindicación 1 independiente adjunta.

Una “máquina virtual” también se denomina una “réplica”. En un aspecto de la invención, se introduce el mecanismo de triplicación y se implementa específicamente por medio de la virtualización. En lugar de tener una triplicación convencional en la que los circuitos de hardware real se triplican, la triplicación de acuerdo con las realizaciones de la invención se virtualizan de una manera especial. Como se desvela, las tres máquinas virtuales se ejecutan en un procesador multinúcleo, usando diferentes núcleos del mismo (es decir, diferentes subpartes del circuito procesador). Cada una de las máquinas virtuales a) se ejecuta en un mismo software de aplicación b) se ejecuta en segmentos de memoria separados y aislados y c) se ejecuta en un núcleo dedicado de un procesador de multinúcleo. En particular, es notable que las tres máquinas virtuales se ejecuten en paralelo. Las máquinas virtuales son máquinas independientes. El término “concurrentemente” puede significar “simultáneamente” en algunas realizaciones, subrayando el aspecto temporal de la ejecución conjunta de las máquinas virtuales. Ya que la ejecución de una aplicación en una máquina virtual es determinista, uno puede esperar obtener los mismos resultados de las tres máquinas virtuales y sustancialmente las mismas duraciones de ejecución para las tres réplicas resultantes de las mismas entradas. Pueden producirse algunos retrasos, debido a la muy compleja ejecución subyacente de las operaciones a nivel de la CPU, por ejemplo, y debido, por ejemplo a la competencia con respecto a los accesos de recursos compartidos a nivel de la memoria. Por ejemplo, las predicciones o eventos de ramificación de la CPU pueden al final diferir ligeramente y una cascada de eventos puede conducir a un cierto retardo en el tiempo. Los umbrales asociados (por ejemplo, dlmáx) se introducen a continuación. El criterio determinista se fuerza con las estimaciones de tiempo de ejecución en el peor de los casos, lo que permite la definición de unos límites de duración máxima. De acuerdo con estos ejemplos, los intervalos (o ventanas de tiempo) se definen (por ejemplo, Wdtime = dlmáx - dlmín) para que distintas partes de una aplicación calculen la misma duración máxima de ejecución para las tres réplicas. Otra forma de expresar esta noción de determinismo de réplica es que cada máquina virtual tiene que reaccionar en el mismo flujo de entrada, de la misma manera, produciendo el mismo flujo de salida dentro del mismo marco de tiempo (al menos sustancialmente para todos estos criterios).

Debe subrayarse la ejecución en paralelo de las tres máquinas virtuales. Los sistemas conocidos en la técnica anterior, por ejemplo, la patente concedida FR2972548, usaron la ejecución secuencial de uno o más programas. Debe subrayarse que la introducción de un mecanismo de virtualización de acuerdo con las realizaciones de la presente invención no es directo per se, por ejemplo, partiendo de este documento FR2972548. Dicho documento desveló tres circuitos de hardware, en lugar de tres instancias de software de acuerdo con algunas realizaciones de la presente invención. A fortiori, la combinación adicional de un mecanismo de triplicación con un mecanismo de virtualización de este tipo es un desafío. Entre muchos aspectos, el mecanismo de triplicación es en su mayoría conocido y dominado por los ingenieros espaciales, mientras que es ampliamente ignorado por los profesionales convencionales de la tecnología de la información (no espaciales). Los últimos expertos están más enfocados en implementar técnicas de virtualización “estándar”. Las realizaciones de la invención desvelan una virtualización específica (que se basa en las técnicas de virtualización “estándar”). Una virtualización específica de este tipo, en

particular con fines de sincronización, plantea problemas técnicos específicos. Por lo tanto, se han introducido mecanismos específicos (por ejemplo, sincronización).

Las tres máquinas virtuales comparten un hipervisor común que puede evaluarse como el denominador común entre los diferentes subsistemas de acuerdo con la invención. Un hipervisor es un programa de software, es decir, unas instrucciones ejecutables por un ordenador. Las réplicas están asociadas con un “comportamiento”, es decir, resultados o salidas determinísticas.

En una realización, el hipervisor puede ser un hipervisor tipo I (1, uno). Este tipo de hipervisor corresponde en general a un programa de software que se ejecuta directamente en la parte superior de la capa de hardware, y en general se denomina hipervisor de “metal desnudo”. Por lo general, es una capa de código fuente delgada con una pequeña superficie de memoria y una sobrecarga de tiempo de ejecución. La ventaja de esta realización es que la pequeña cantidad de código implicada puede hacer que la solución global sea menos propensa a errores. En otra realización, el hipervisor puede ser un hipervisor tipo II (2, dos). Este tipo de hipervisor corresponde a un programa de software que depende de un sistema operativo (SO) y en general se denomina hipervisor tipo II. Dichos hipervisores proporcionan en general características más ricas, en general son menos específicos del hardware pero también en general conducen a un tiempo de ejecución más largo. La cantidad de código de software involucrado suele ser más importante (pero, por ejemplo, las distribuciones ligeras del sistema operativo que soportan la virtualización están disponibles y pueden difuminar la diferencia). En vista de las evoluciones objetivas y previsibles de la arquitectura de los procesadores de hardware, un hipervisor puede incorporarse como un firmware o un circuito integrado digital, que puede ser más eficiente (por ejemplo, más rápido) que sus equivalentes de software.

Un fallo puede ser un mensaje de error en el flujo del mensaje. Un comportamiento defectuoso puede asociarse con o resultar en un evento SEFI (interrupción) u otra anormalidad. Tras la detección del fallo, pueden producirse algunas operaciones de diagnóstico y reacción. Puede evaluarse un fallo en la duración esperada para producir una salida por una máquina virtual.

La máquina virtual que ha producido un fallo se califica como una máquina virtual “defectuosa”. El fallo se evalúa mediante comparaciones de los resultados de salida de las máquinas virtuales (deterministas).

El software de aplicación se replica en un elemento informático (CE), - esta expresión encapsula tanto el núcleo de procesador como el segmento de memoria – ejecutando cada una de estas subpartes de circuitos en una réplica o una máquina virtual, cada uno ejecutándose en un núcleo de procesador dedicado, y ejecutándose en segmentos de memoria aislados con enlaces definidos de comunicación entre particiones (IPC). Estas réplicas procesan el mismo contexto de entrada de datos en un tiempo limitado y producen los mensajes de salida (valores de datos) para una capa de software de comunicación y sincronización (CS) del marco. Esta capa interconecta la carga útil del software de aplicación (ASW) (espacio) y sincroniza todas las réplicas y sus valores de salida de datos o mensajes de salida (por ejemplo, numeración de secuencia, marca de tiempo global), los estructura (por ejemplo encabezado, carga útil, suma de comprobación, campos de finalizador) siguiendo un protocolo de comunicación y agrega una extensión de seguridad (también de acuerdo con el número de réplica en que reside) específica al campo de finalizador con el fin de obtener un mensaje confiable.

En un desarrollo, el procedimiento comprende además la etapa de ejecutar un voto distribuido en los mensajes de salida de las máquinas virtuales para determinar un mensaje de salida votado.

En el nivel más alto de abstracción, el procedimiento comprende una etapa de voto distribuido. Un procedimiento de este tipo (y sus variantes) se conoce independientemente en la técnica, pero su combinación con las realizaciones de la invención no lo es. En un aspecto, los resultados de datos de salida del software de aplicación (ASW) en primer lugar no se transmiten fuera de todas las máquinas virtuales (plataforma informática asociada (CP)). Pero se difunden a todos los elementos informáticos (CE significa réplicas) dentro de la plataforma informática CP. Para un sistema externo a la CP, los CE o réplicas actúan como una sola entidad, es decir, no son “visibles”, es decir, la CP es el único sistema global con el que interactuar. Los CE/réplicas son componentes internos. Cada partición tiene un mensaje único local con esos datos listos para enviarse a las réplicas vecinas para su votación. La capa de software de comunicación y sincronización (CS) de cada réplica trata con todas las transmisiones de mensajes punto a punto de las interparticiones (por ejemplo, redundancia de conexiones, control de tiempo y protocolos) basándose en una pila de comunicación local.

En un desarrollo, cada máquina virtual extrae o inserta un mensaje de salida de/en las otras máquinas virtuales.

En este desarrollo específico, cada partición comienza los intercambios en un estado emisor para insertar su mensaje local en todas las particiones (también conocido como la réplica de sí mismo) a través de cada enlace IPC específico. A continuación, para continuar los intercambios, cada partición cambia al estado de receptor para extraer todos los mensajes recibidos de todas las particiones a través de sus enlaces. En otras palabras, los resultados de datos de salida del software de aplicación (ASW) aún están dentro de la plataforma informática. Cada partición tiene las tres instancias de esos datos incorporados en tres mensajes grabados en una caja de depósito listos para votar.

La capa de software de votación del marco está distribuida alrededor de todas las particiones. En otras palabras, cada partición incorpora una instancia del votante que es capaz de llegar a un acuerdo y confirmar datos por sí

solos. El voto consiste esencialmente en funciones de comparación; una para la parte especificada de datos de todo el mensaje (por ejemplo, una comparación de bytes a bytes) y otra que analiza la integridad del código de extensión de seguridad parcial. Finalmente, una función de votación construye un mensaje votado con la extensión de seguridad completa si se han identificado al menos dos mensajes correctos.

En un desarrollo adicional, los mensajes de salida de cada máquina virtual se recopilan en una caja de depósito y un mensaje de salida votado se determina fuera de los mensajes de salida. El voto distribuido se realiza en dos subetapas. En un desarrollo, el voto distribuido se realiza después de una etapa anterior de recopilar los mensajes de votación de cada réplica en una caja de depósito antes de comenzar el procedimiento de votación. En otras palabras, cada votante también se triplica. Desde ahora, todos los mensajes de las réplicas están a disposición de cada partición en su caja de depósito y, por lo tanto, todas las instancias replicadas del votante pueden votar por su propia cuenta. Esto significa que se realizan tres votos en paralelo en diferentes núcleos, usando los mismos mensajes redundantes pero localizados en diferentes regiones de memoria.

En un desarrollo, el procedimiento comprende además la etapa de comunicar el mensaje de salida votado a un sistema externo en comunicación con el sistema.

Como se ha tratado, para un sistema externo en interacción con la unidad de circuito de hardware o CP, las réplicas actúan como una sola entidad.

En un desarrollo, cada máquina virtual está predefinida como primaria o como secundaria, y en el que el mensaje de salida votado se comunica por la máquina virtual primaria o por la máquina virtual secundaria si la máquina virtual primaria es defectuosa.

En este punto en el tiempo, cada partición tiene un mensaje votado con mensajes o resultados de datos de salida del software de aplicación (ASW) listos para enviarse fuera de la plataforma informática. Con el fin de decidir qué partición lo enviará, por configuración, una partición se define como primaria (también conocida como remitente maestro si no tiene fallos) y otra como secundaria (en caso de una partición primaria defectuosa).

En un desarrollo, los mensajes de salida de las máquinas virtuales están numeradas y/o con marcas de tiempo y/o estructuradas y/o anotadas.

En este desarrollo, los metadatos (“datos sobre los datos”) se añaden a los datos. Por ejemplo, los datos pueden corresponder a la carga útil de aplicación y los metadatos pueden corresponder a una extensión de seguridad (por ejemplo, un código de acuerdo con el número de réplica en el que reside la aplicación). Con respecto a los datos, los mensajes de salida también pueden estructurarse o reestructurarse (por ejemplo, con encabezado, carga útil, suma de comprobación, campos de finalizador), por ejemplo, siguiendo un protocolo de comunicación.

Con respecto a los metadatos, los valores de salida de datos o mensajes de salida de las réplicas o máquinas virtuales pueden numerarse u ordenarse o marcarse en el tiempo (por ejemplo, numeración de secuencia, marca de tiempo global). Además, los mensajes de salida pueden anotarse (metadatos), es decir, una extensión o anotación de seguridad específica puede agregarse al campo de finalizador o asociarse con los mensajes de salida (también conocido como el número de réplica en el que reside). El efecto técnico (consecuencia) es que se obtienen mensajes de salida confiables.

En un desarrollo, el procedimiento de la reivindicación anterior comprende además la etapa de asociar una extensión de seguridad con un mensaje de salida de una máquina virtual, comprendiendo dicha extensión de seguridad una información de identificación sobre la máquina virtual que emite el mensaje de salida.

En una realización específica, el procedimiento comprende además la etapa de agregar o asociar una extensión de seguridad como metadatos. Esta extensión de seguridad, por ejemplo, puede ser la desvelada en el documento EP0977395 titulado “Method of secure monochannel transfer of data between nodes of a network, computer network and computer nodes”. La unión de seguridad en una realización es una extensión realizada por el emisor de un mensaje que permite al receptor detectar cambios intencionados o no intencionados incorporados al mensaje durante la transferencia del mensaje. En una realización, un canal de cálculo (por ejemplo, el emisor primario o secundario) debe preferentemente no ser capaz de calcular un mensaje válido por sí solo. Para el cálculo de un mensaje válido, se deben involucrar al menos dos particiones para generar un mensaje válido. Para esto, en una realización, con un procedimiento de construcción dedicado, puede anexarse una extensión de seguridad a los mensajes.

En una realización, la extensión de seguridad presenta una longitud configurada fija (de cuatro hasta diez bytes) que se añade a los metadatos asociados con un mensaje. Puede usarse un procedimiento definido para evitar que cada votante calcule un mensaje válido por sí mismo. Cada aplicación agrega un apéndice de seguridad dañado al mensaje omitiendo una parte del código de seguridad correspondiente a su localización. Puede anexarse un apéndice de seguridad sin el n-ésimo byte de acuerdo con el número de réplica en el que reside. Todas las instancias del votante pueden obtener los mensajes de todas las particiones (es decir, de la partición del propio votante y de las particiones vecinas) con estas firmas de seguridad parciales. Los votantes pueden montar el mensaje completo completando las partes omitidas de las particiones individuales. Con el fin de construir el código

de seguridad completo, cada votante puede usar dos mensajes con diferentes firmas de seguridad (es decir, mensajes provistos de diferentes particiones). Siempre que el votante haya identificado dos mensajes de acuerdo con los datos de usuario y la extensión de seguridad (parcial) para que sea correcta, puede construir un mensaje votado con la extensión de seguridad completa. Como resultado, cada partición tiene un mensaje válido (es decir, votado) con la longitud correcta.

En un desarrollo, el procedimiento comprende además la etapa de recuperar la máquina virtual defectuosa. La recuperación puede manejarse de varias maneras.

En un desarrollo, el procedimiento comprende además una etapa de recuperación de la máquina virtual defectuosa, etapa que comprende usar el contexto de datos de una máquina virtual no defectuosa para reemplazar el contexto de datos de la máquina virtual defectuosa.

En este desarrollo, los datos de contexto de la aplicación se reinyecta, por ejemplo, para reiniciar una máquina virtual. Por lo tanto, el contexto de datos o datos de contexto comprende datos acerca de la propia máquina virtual (por ejemplo asignación de RAM, direcciones IP, etc.) y también datos acerca de la ASW de aplicación (por ejemplo, valores de ciertas variables, estados de ventanas de GUI, memoria intermedia de estado de eventos, etc.). Como pueden hacerse copias de seguridad de memorias intermedias o memorias temporales o memorias caché, puede recuperarse en cierta medida un estado anterior (o “snapshot”) en el pasado reciente y restablecer los estados de la máquina virtual y de la aplicación (al menos hasta un punto anterior en el tiempo antes del punto de error). Notablemente, tales datos de contexto son finitos, es decir, hay un conjunto definido de parámetros (y no infinito) que restaurar para recuperarse del error. Los experimentos y prototipos han demostrado que pueden manejarse un par de decenas de parámetros. En una realización, dicho contexto de datos puede calificarse mediante el “contexto de datos funcional”.

En un desarrollo, la etapa de recuperar la máquina virtual defectuosa se realiza en un punto de resincronización en el tiempo.

La estrategia de recuperación propuesta en el presente documento puede usar una sincronización operativa entre los procedimientos replicados que funcionan en paralelo. Las máquinas virtuales se ejecutan en paralelo y la sincronización puede manejarse en algunos puntos de sincronización, predefinidos o no. En una realización, el software ejecutado en paralelo dentro de las máquinas virtuales puede dividirse en intervalos sincronizados. La partición en intervalos puede proporcionar los puntos de sincronización donde se inician las partes (es decir, difusión de mensajes, votación, recuperación) del algoritmo.

En un desarrollo, la ejecución se corta gracias un evento de hardware externo que proporciona puntos de sincronización por eventos.

Una realización para ejecutar una partición periódicamente es, por ejemplo, un temporizador que proporciona puntos de sincronización de activación por tiempo. Uno de estos puntos de sincronización desencadena, en los procedimientos replicados que funcionan simultáneamente en cada máquina virtual, el intervalo para la ejecución de la estrategia de recuperación.

Se subraya que los puntos de sincronización por tiempo son opcionales (no se necesitan). Corresponden a una realización específica. En algunas realizaciones, puede implementarse la sincronización por “latido” (es decir, la sincronización se produce a intervalos de tiempo definidos, y no necesariamente de manera regular). En otras realizaciones adicionales, la sincronización puede “activarse por eventos” (es decir, unos eventos específicos pueden desencadenar una o más sincronizaciones entre máquinas virtuales). En otras palabras, un “reloj” o un “temporizador” (y similares) son totalmente opcionales y no son absolutamente necesarios.

En un desarrollo, se asocia un fallo con un error elegido de la lista que comprende: error de caída, error de valor defectuoso, error bizantino, error de temporizador y combinaciones de los mismos. En una realización, un votante opera únicamente en flujos de mensajes. Vota sobre mensajes redundantes e intenta encontrar un acuerdo (es decir, votar sobre estos mensajes). Todos los comportamientos defectuosos de una réplica determinada se manifiestan a través de anomalías en la secuencia de mensajes. Un votante puede detectar estos mensajes inusuales y puede informar a la capa del administrador de fallos. El votante diagnostica errores en el flujo de mensajes basándose en detecciones durante ventanas de tiempo predefinidas. El fallo “error de caída” designa una réplica que ya no envía mensajes, enlaces entre dos réplicas o partes de envío en una réplica que están rotas (ya no funcionan), un error de software/hardware que lleva al envío de mensajes ilegales en una réplica (las otras réplicas tienen que tratar con estos mensajes erróneos). Tales errores se manifiestan ellos mismos en fallos que pueden ser permanentes o transitorios. Los fallos permanentes son fallos que influyen en la transmisión de mensajes de manera permanente. Los fallos transitorios son fallos que influyen en la transmisión de mensajes dentro de un tiempo dado y se miden en porcentaje.

Se desvela un programa informático que comprende unas instrucciones para realizar una cualquiera de las etapas del procedimiento cuando dicho programa informático se ejecuta en un dispositivo informático adecuado. Se desvela un medio legible por ordenador que tiene codificado en el mismo un programa informático de este tipo. También se desvela un sistema que comprende medios adaptados para realizar una cualquiera de las etapas del procedimiento.

Descripción de los dibujos

Las realizaciones de la presente invención se describirán ahora a modo de ejemplo haciendo referencia a los dibujos adjuntos en los que las referencias indican elementos similares, y en los que:

la figura 1 ilustra una vista de sistema de una realización a modo de ejemplo de la arquitectura de marco tolerante a fallos (FT-Fwk) con triple redundancia de software (TSwR);

la figura 2 detalla la vista de sistema del marco de software;

la figura 3 ilustra un ejemplo de una secuencia de comunicación y sincronización;

la figura 4 ilustra la línea de tiempo de un escenario a modo de ejemplo dinámico, nominal y defectuoso.

Descripción detallada de la invención

Se utilizan los siguientes acrónimos: componente sacado del estante (COTS); alteración por evento único (SEU); interrupciones funcionales por evento único (SEFI); triple redundancia modular triple (TMR); software de aplicación (ASW); elementos informáticos (CE); plataforma informática (CP); triple redundancia de software (TSwR); arquitectura de marco tolerante a fallos (FT-Fwk); unidad de datos de carga útil (PDU).

Una alteración por evento único (SEU) corresponde a un cambio en un estado de un bit (un elemento elemental de información) dentro del procesador provocado por una partícula, por ejemplo, un ion pesado.

Una interrupción funcional por evento único (SEFI) corresponde a un estado de bloqueo del procesador. Este evento puede ser una consecuencia directa de una alteración por evento único (SEU) que ha provocado un cambio en el comportamiento del procesador.

Una plataforma informática (CP) es una máquina de hardware que tiene instaladas en el procesador multinúcleo COTS unidades de memoria (por ejemplo, basadas en RAM, EEPROM o PROM) e interfaces de entrada/salida de datos (I/O) (por ejemplo, Ethernet, CAN, buses I2C).

Unos elementos informáticos (CE) son máquinas virtuales (también conocidas como particiones) con un conjunto de recursos de hardware virtual tal como un núcleo de procesamiento, unos segmentos de memoria aislados y un subconjunto de interfaces de entrada/salida basados en el particionamiento de recursos de hardware de plataforma informática (CP).

Los ejemplos proporcionados desvelan un enfoque a nivel de sistema con una arquitectura de SW (software)/HW (hardware) combinada para realizar la mitigación de sensibilidad a la radiación de la plataforma informática.

Las realizaciones de los procedimientos y sistemas develan una arquitectura de marco tolerante a fallos (FT-Fwk) con triple redundancia de software (TSwR) para operar el software de aplicación de carga útil de satélite (ASW) en un procesador multinúcleo COTS moderno con tecnología de virtualización y memoria protegida EDAC (es decir, mecanismos de ECC - código de corrección de errores).

De acuerdo con algunas realizaciones, se implementa una redundancia de triple software con máquinas virtuales que trabajan en segmentos de memoria separados (y aislados) y simultáneamente en la parte superior de una capa de software de hipervisor en lugar de una solución TMR clásica. Para una mayor disponibilidad (y, por lo tanto, confiabilidad), las salidas de software de aplicación por triplicado (ASW) se votan con un procedimiento de votación de seguridad distribuida.

En un desarrollo, el “votante” se “distribuye” (es decir, a lo largo de las tres máquinas virtuales) con el fin de minimizar el punto único de software de error. Este votante o estos votantes operan con mensajes redundantes (es decir, intentan encontrar un acuerdo) con la capacidad de detección instantánea de errores que se manifiestan en los recursos muy utilizados. Los errores pueden corresponderse con comportamientos defectuosos resultantes de eventos SEU (alteración por evento único) que se manifiestan en anomalías en la secuencia de mensajes (por ejemplo, “el mensaje no se encuentra” o “el contenido del mensaje es incorrecto”) se detectan y enmascaran.

De acuerdo con otro desarrollo, se desvela una monitorización “sincronizada” de réplicas que opera en un comportamiento determinista de las réplicas; las réplicas no defectuosas proporcionan un comportamiento de salida idéntico dentro de un intervalo de tiempo dado (por ejemplo, incluso aumentado, tal como el tiempo de ejecución de peor caso - WCET). En este extremo, la invención hace que los mecanismos dentro del marco y la extensión del software de aplicación (ASW) para la aplicación del determinismo de las réplicas cubran el comportamiento potencial no determinista de la arquitectura de los procesadores modernos. Por lo tanto, solo los comportamientos defectuosos resultantes de eventos SEFI que se manifiestan en anomalías en la duración para producir mensajes (*como ejemplos: expiración del tiempo de espera de sincronización de mensajes o no más respuestas de las réplicas*) se detectan, se aíslan y se recuperarán.

En algunas realizaciones, los procedimientos y sistemas comprenden una redundancia de triple software (también

conocida como lógico en lugar de físico) de elementos informáticos (CE) dentro de la arquitectura de marco tolerante a errores que ejecuta una carga útil de software de aplicación espacial (ASW) tres veces en una plataforma informática (CP). El principio es usar máquinas virtuales (VM) como CE sobre una plataforma informática (CP) basada en procesadores multinúcleo comercial en lugar de diversos CE físicos para realizar una redundancia de triplicación del software de aplicación (ASW).

En algunas realizaciones, se ejecutan tres réplicas sincronizadas de una carga útil de software de aplicación espacial (ASW). Cada ejecución se monitoriza para la detección de fallos. Las réplicas defectuosas se recuperan y las réplicas de votos se emiten con un procedimiento de votación por mayoría de seguridad distribuida.

Una capa de software de hipervisor se usa para predefinir las máquinas virtuales para funcionar a partir de segmentos de memoria separados y aislados y para operar simultáneamente en diferentes núcleos de procesamiento. Ambas ventajas reducen las probabilidades de situación de bloqueo de carga útil de aplicación.

Se define un marco de software distribuido para implementar la sincronización, la comunicación y funcionalidades tolerantes a fallos habituales, tales como la gestión de fallos (por ejemplo, detección de fallos, diagnóstico y reacción). Este enfoque contribuye a reducir las probabilidades de situación de bloqueo minimizando el punto único de software de error.

En un desarrollo, se desvela un procedimiento de votación distribuida (también conocido como tres casos de votante mayoritarios “2 de cada 3”) que confirma y encuentra un acuerdo entre los mensajes de salida de software de aplicación (ASW) extendidos con los defectos de protocolo de seguridad tales como la cifra de ronda, la marca de tiempo global y el código de extensión de seguridad único. Este enfoque contribuye a mejorar la disponibilidad de carga útil de aplicación y a minimizar el punto único de software de error.

En un desarrollo, se desvela una estrategia de recuperación (y las etapas asociadas). La recuperación es un tipo de recuperación de datos redundante que usa el contexto de datos de una réplica válida para reemplazar el contexto de datos incorrecto de la réplica errónea en un punto en el tiempo (también conocido como la estrategia de recuperación que usa una sincronización operativa entre procedimientos replicados que trabajan en paralelo).

La invención opera en una plataforma informática que comprende un procesador multinúcleo moderno comercial, unidades de memoria e interfaces de entrada/salida de datos. Otro aspecto, en el que se basa el marco tolerante a fallos, es una capa de software de hipervisor que permite la ejecución simultánea de varias MV en esta plataforma informática (CP).

El software de aplicación (ASW) se replica en cada elemento informático (CE), conteniendo cada uno un núcleo de procesamiento dedicado, segmentos de memoria aislados y enlaces definidos de comunicación entre particiones (IPC). Estas réplicas procesan el mismo contexto de entrada de datos en un tiempo limitado y producen los valores de salida de datos para una capa de software de comunicación y sincronización (CS) del marco. Esta capa interconecta la carga útil de software de aplicación espacial (ASW) y sincroniza todas las réplicas y sus valores de salida de datos (por ejemplo, numeración de secuencia, marca de tiempo global), sus estructuras (por ejemplo encabezado, carga útil, suma de comprobación, campos de finalizador) que siguen un protocolo de comunicación y añade una extensión de seguridad específica (también de acuerdo con el número de réplica en la que reside) en el campo de finalizador para obtener un mensaje confiable.

En este punto en el tiempo, los resultados de los datos de salida del software de aplicación (ASW) no se transmiten fuera de la plataforma informática CP. Cada partición tiene un mensaje único local con esos datos listos para enviarse a las réplicas vecinas para su votación.

La capa de software CS de cada réplica trata con las transmisiones de mensajes punto a punto todas las interparticiones (por ejemplo, la redundancia de conexiones, control de tiempo y protocolos) basándose en una pila de comunicación local. Cada partición inicia los intercambios en un estado emisor para insertar su mensaje local en todas las particiones (también incluyendo la propia réplica) a través de cada enlace IPC específico. A continuación, para continuar los intercambios, cada partición cambia al estado de receptor para extraer todos los mensajes recibidos de todas las particiones a través de sus enlaces.

En este punto en el tiempo, los resultados de los datos de salida del software de aplicación (ASW) están aún dentro de la plataforma informática. Cada partición tiene las tres instancias de esos datos incorporados en tres mensajes grabados en una caja de depósito lista para votar.

La capa de software de votación del marco se distribuye entre todas las particiones. En otras palabras, cada partición incorpora una instancia del votante que es capaz de encontrar un acuerdo y confirmar solo los datos. El voto consiste esencialmente en unas funciones de comparación; una para la parte especificada de datos de todo el mensaje (por ejemplo, una comparación de bytes a bytes) y otra que analiza la integridad del código de extensión de seguridad parcial. Finalmente, una función de votación construye un mensaje votado con la extensión de seguridad completa si se han identificado al menos dos mensajes correctos.

En este punto en el tiempo, cada partición tiene un mensaje votado con los resultados de los datos de salida del

software de aplicación (ASW) listos para enviarse fuera de la plataforma informática. Con el fin de decidir qué partición lo enviará, por configuración, una partición se define como primaria (también conocida como remitente maestro si no tiene fallos) y otra como secundaria (en el caso de una partición primaria defectuosa).

En algunas realizaciones, los procesadores multinúcleo modernos comercialmente disponibles pueden usarse para aplicaciones espaciales (por ejemplo, PowerPC o procesadores de señal digital DSP). En particular, los efectos SEU y SEFI pueden gestionarse manejando los siguientes errores:

- error de caída: un SEU/SEFI conduce o bien a una réplica que no envía más mensajes; o ambos enlaces entre dos réplicas están rotos o una réplica de envío está caída/bloqueada;
- error de valor defectuoso: un SEU conduce al envío de mensajes ilegales en una réplica;
- error bizantino: una réplica envía un mensaje diferente (pero autenticado) a las réplicas vecinas en la misma ronda;
- error de temporizador: una réplica inicia un nuevo punto de sincronización antes de que el período mínimo de la ronda haya expirado.

En el caso de que se produzca uno de estos errores, puede usarse una capa de software de gestión de errores a cargo de la diagnosis de fallo y de la reacción a fallos basándose en la información de detección de errores reportada por diferentes componentes de marco.

Una estrategia de recuperación puede usarse para reintegrar una réplica defectuosa durante la operación sin interrupción del servicio: por ejemplo, la réplica defectuosa puede recargarse y reiniciarse, con el contexto de datos de entrada desde una réplica correcta (mientras, por ejemplo, las réplicas vecinas no progresan)
La figura 1 ilustra una vista de sistema de una realización a modo de ejemplo de la arquitectura de marco tolerante a fallos (FT-Fwk) con triple redundancia de software (TSwR), con un procesador multinúcleo apropiadamente seleccionado y configurado sabiamente:
El hardware de plataforma informática (CP) y sus recursos físicos particionados comprenden una pluralidad de núcleos de procesamiento (por ejemplo, #1.1, #2.1 y #3.1); una pluralidad de segmentos de memoria basados en SDRAM (memoria de acceso aleatorio dinámico síncrono) (por ejemplo, #1.2, #2.2 y #3.2); y una pluralidad de canales de memoria compartida (por ejemplo, #12, #23, #31) y una interfaz de E/S (por ejemplo, #4.1).
La configuración de tres unidades es solo un ejemplo. Una plataforma informática puede comprender un mayor número de unidades (al menos 3 y por ejemplo 186 unidades en configuración de réplica, o cualquier otro número, en correspondencia con la cantidad de núcleos de procesadores modernos, cuyos mapas de ruta incluyen procesadores de 1000 núcleos o incluso más).
Por ejemplo, en otra realización, el recurso de procesamiento puede ser un procesador de cuatro núcleos (con un núcleo deshabilitado o forzado al estado inactivo) con características de virtualización de hardware (por ejemplo, un modo de privilegio adicional de ejecución a nivel de procesador), con uno o dos niveles de memoria caché que incluyen mecanismos de protección (por ejemplo, bits de paridad, código de corrección de errores - ECC) y con una unidad de gestión de memoria (MMU) que segmenta el espacio de memoria direccionable.

La capa #4 de software de hipervisor (HV), elegida apropiadamente, crea y gestiona los CE, asigna los recursos físicos para los mismos y garantiza esta asignación y su acceso durante el tiempo de ejecución.

En una variante, el hipervisor #4 es una capa de software delgada con servicios para gestionar eventos a nivel de procesador y en particular las interrupciones, funcionando en el modo de privilegio más alto de ejecución y ofreciendo interfaces abiertas (API) para las extensiones ad-hoc.

Los elementos informáticos #1, #2 y #3 son las particiones, definidas con un conjunto dedicado de recursos de hardware virtuales basados en el particionamiento de los recursos de hardware reales y en el funcionamiento de las réplicas #10, #20 y #30 de la carga útil de software de aplicación (ASW) conectadas correctamente a las instancias #100, #200 y #300 del marco tolerante a fallos (FT-Fwk) con triple redundancia de software (TSwR). En el ejemplo, el CE #3 se configura como el canal primario para las transmisiones externas y el CE #1 se configura como un canal secundario en el caso de que el CE #3 se haya detectado como defectuoso. El CE #2 no necesita necesariamente un canal de transmisión externo.

Una realización de ejemplo de una carga útil de software de aplicación (ASW) se describe en lo sucesivo en el presente documento para las réplicas #10, #20 y #30. Se presentan los principios de comportamiento e interfaz de esta aplicación de carga útil necesarios para la comprensión de la invención:

- la aplicación de carga útil se lanza con un contexto de entrada de datos de inicio y ejecuta rondas informáticas para siempre, actualizando cada vez el contexto de entrada de datos, para producir valores de salida de datos correspondientes (para el exterior). La duración del tiempo de una ronda de cálculo es finita y puede limitarse.
- la entrada de datos proviene de un segmento de memoria (no mostrado) identificado por una dirección base conocida y un desplazamiento de dirección, es decir, el contexto. Pueden preverse otras variantes de datos entrantes, tales como los datos periódicos desencadenados por tiempo, procedentes de una interfaz de entrada/salida externa.
- una ronda de cálculo consiste en procesar la entrada de datos del contexto de entrada, para producir los valores

de salida de datos correspondientes al exterior y para actualizar el contexto de entrada para la siguiente ronda de cálculo. En el caso de la entrada de datos desencadenados por tiempo, el contexto se reemplaza por un período de tiempo síncrono.

- los valores de salida de datos que son los resultados de aplicación de carga útil se transmiten por un canal que se comunica con el exterior a través de la interfaz de entrada/salida. Pueden usarse varias variantes de esta interfaz, tales como los buses Ethernet, I2C o SPI.

La figura 2 detalla la vista del sistema del marco de software.

La figura comprende una interfaz 200 de aplicación de tiempo de ejecución, un componente 210 de monitorización de estado, un componente 220 de comunicación y sincronización, un votante 230 distribuido, un componente 240 de gestión de recuperación y un componente 250 de gestión de fallos.

Un componente 200 de aplicación de tiempo de ejecución es un componente de aplicación a cargo de hacer funcionar el ASW de una manera replicable a través de interfaces bien definidas que minimicen las uniones de ASW al marco TSwR y gestionar el canal virtual (equivalente al marco OCS). Como ejemplo, esto abre el canal virtual y establece todas las conexiones de subcanales subyacentes. “Aplicación” identifica un grupo de componentes del marco TSwR que interactúa con el dominio ASW y funciona en el nivel de usuario (en oposición al superusuario y al hipervisor).

El componente 210 de monitorización de estado es un componente a cargo de la implementación de mecanismos de tiempo de ejecución apropiados para controlar la seguridad del sistema. Como ejemplo, monitoriza la ejecución del hipervisor y las réplicas (por ejemplo, tiempos de espera, guardián).

El componente 220 de comunicación y sincronización es un componente de núcleo a cargo de diversos aspectos tales como los servicios de colas de mensajes, la sincronía del funcionamiento ASW y la distribución de mensajes alrededor de todas las réplicas. “Núcleo” identifica una capa clave del marco TSwR para garantizar el comportamiento determinista y la coherencia de los mensajes.

Un componente 230 de votante distribuido es un componente repartido alrededor de todas las réplicas que se encarga de encontrar un acuerdo sobre los mensajes redundantes de ASW replicado. Los componentes de software de este bloque están estrechamente acoplados con la capa de sincronización y el gestor de fallos.

Un componente 240 de gestión de recuperación es un componente a cargo de reintegrar una réplica ASW defectuosa durante el funcionamiento sin interrupción del servicio. Como ejemplo, transfiere los estados y la información de datos desde una réplica correcta mientras la aplicación no realiza ningún progreso.

Un componente 250 de gestión de fallos es un componente de núcleo a cargo de la diagnosis de fallos y la reacción a fallos basándose en la información de detección de fallos reportada por los diferentes componentes de marco. Es responsable de ejecutar la reacción apropiada ante fallos y manejar la gestión de redundancia. “Núcleo” identifica un componente clave del marco TSwR para lograr la necesidad de tolerancia a fallos.

La figura ilustra un esquema funcional del marco de software, que comprende componentes funcionales, mecanismos específicos y bloques de construcción de software. La figura muestra un elemento informático CE #x (con su hardware virtual), donde x es un número de un CE, un software de aplicación (ASW) #x0 y un marco FT-Fwk #x00. Este esquema funcional de la invención puede aplicarse a cualquier elemento informático CE, software de aplicación (ASW) y FT-Fwk de la arquitectura de software. La superposición del Hipervisor #4 y el FT-Fwk #x00 ilustra que pueden implementarse diversas variantes de la arquitectura de ruptura estática de software.

Una interfaz 200 de aplicación de tiempo de ejecución es un componente que puede estar a cargo de hacer funcionar la carga útil de software de aplicación (ASW) de una manera replicable a través de las interfaces de software definidas que minimizan las uniones de carga útil de software de aplicación (ASW) a la arquitectura TSwR FT-FWK. Para este fin, el componente proporciona, por un lado, una interfaz como punto de entrada para lanzar el software de aplicación (ASW) con el contexto de entrada de datos sincronizado (y correcto), y proporciona por otro lado una interfaz de punto de sincronización que se llamará por el software de aplicación (ASW) al final de cada ronda de cálculo con los valores de salida de datos producidos. La última interfaz formatea los valores de salida de datos producidos y la información de contexto de ronda en un mensaje de unidad de datos de carga útil (PDU) para pasar a la capa de comunicación y sincronización.

El componente 210 de monitorización de estado es un componente funcional a cargo de la implementación de los mecanismos de tiempo de ejecución apropiados para controlar las operaciones de software y del procesador. De acuerdo con esta característica tolerante a fallos, el procesador elegido proporciona un mecanismo de guardián para cada núcleo configurado y gestionado a nivel de privilegios de hipervisor. Para fines de detección de fallos, el componente monitoriza la ejecución del hipervisor y de las réplicas (por ejemplo, expiración del tiempo de espera, guardián). En este extremo, se proporciona una interfaz para hacer ping, a un intervalo regular, al guardián para notificarle que todo está funcionando correctamente. En ausencia de dicha llamada de ping al final de un período de tiempo predefinido, el guardián reinicia el núcleo (o respectivamente el procesador) ejecutando el software de elemento informático CE #x (o respectivamente el hipervisor #4). Mediante este mecanismo, se detecta un estado de

bloqueo de hardware y/o software y puede rectificarse. En algunas realizaciones, este componente puede comprender medios (e interfaces correspondientes) para calcular retrasos y desencadenar condiciones de expiración del tiempo de espera (por ejemplo, implementación de alarmas o temporizadores - controlador/indicadores de inicio/parada/interrupción). En otra realización asociada con una función de guardián de monitorización de estado, se usa un mecanismo de guardián de hardware externo adicional para activar el reinicio del procesador. El hipervisor envía, a intervalos regulares, una señal a este mecanismo externo para notificarle que está funcionando correctamente. En ausencia de una señal de este tipo al final de un período de tiempo predefinido, el guardián de hardware externo reinicia el procesador.

El componente 220 de comunicación y sincronización ("un módulo de globalización es un componente que está principalmente a cargo de la sincronización entre las réplicas (#10, #20 y #30) de software de aplicación (ASW), y la distribución de los valores de salida de datos redundantes de réplicas de software de aplicación (ASW) alrededor de todas las instancias de FT-Fwk (#100, #200 y #300). La plataforma informática (CP) está asociada con una configuración replicada. Las instancias de réplica del software de aplicación (ASW) están sincronizadas. Con el fin de definir los estados en que pueden correlacionarse valores de salida redundantes con respecto a órdenes de mensajes y contenido de datos, la sincronización tiene lugar cuando el software de aplicación (ASW) utiliza interfaces de software definidas, creando puntos de bloqueo síncronos (también útiles para una reintegración de réplica defectuosa). La detección de fallo se realiza por comparación de los comportamientos de salida que incluyen el comportamiento de envío de mensajes de las réplicas del software de aplicación (ASW) (a través de la interfaz 200 de aplicación de tiempo de ejecución). El servicio central del módulo de capa de sincronización del FT-Fwk es la "globalización" de los mensajes. Este servicio define qué mensajes, disponibles en un CE, se distribuyen. El servicio distribuye mensajes de tal manera que estos mensajes estén disponibles de la misma manera consistente en todos los CE en la plataforma informática (CP). Para este fin, la estrategia para invocar dicha actividad de globalización es permitir que cada entidad de réplica inicie una actividad de sincronización con el punto de bloqueo síncrono. Todas las réplicas de socios pueden responder inmediatamente a la actividad de sincronización iniciada. En otras palabras, la primera réplica que inicia la globalización establece un retraso definido (por ejemplo, usando unos servicios de monitorización de estado) y todas las réplicas restantes tienen que responder a la actividad de globalización iniciada antes de la expiración del tiempo de espera. En ausencia de respuesta de una réplica, la monitorización de estado lo detecta y esta réplica se marca como "defectuosa" por la gestión de fallos. En algunas realizaciones, este módulo de globalización comprende medios para construir un "tiempo global" de plataforma informática (CP) para marcar el tiempo de mensajes que se calcula con réplicas de tiempo local sin errores (en este punto en el tiempo). Son posibles varias variantes de implementación equivalentes (por ejemplo, calculando el valor promedio o el valor mediano). En algunas realizaciones, este componente comprende medios para intercambiar datos relevantes para la seguridad a través de diferentes medios de transmisión existentes (por ejemplo, memoria compartida, Ethernet, interfaz de E/S...). El módulo de capa de comunicación de cada FT-Fwk tiene una pila de comunicación individual con una capa de protocolo, una capa de integridad de datos y una capa de dispositivo. La capa de protocolo es responsable del control y la monitorización del procedimiento de transferencia de mensajes, mientras que la capa de integridad de datos proporciona el examen de mensajes recibidos y la construcción de los mensajes para su emisión, incluido el código de extensión de seguridad único. La tarea de la capa del dispositivo consiste esencialmente en estructurar y monitorizar la finalización del intercambio de datos de acuerdo con los requisitos de los casos de medios de transmisión existentes.

Un votante (231, 232, 233) distribuido es un componente que está a cargo de la búsqueda de un acuerdo sobre los mensajes redundantes de los elementos informáticos (CE #1, CE #2 y CE #3), determinando y confirmando el mensaje correcto fuera de la plataforma informática (CP). Cada votante se instancia en cada réplica (dentro de #100, #200 y #300), asocia los mensajes correspondientes y coloca este triplete en una caja de depósito.

Un módulo (231, 232, 233) (capa) de votante está estrechamente acoplado con el componente 250 de gestión de fallos por razones de presentación de informes de error y con el componente 220 de comunicación y sincronización para obtener los mensajes recibidos en una caja de depósito y para transmitir también al exterior el mensaje votado.

Un componente 240 de gestor de recuperación es un componente que está a cargo de reintegrar una réplica defectuosa durante la operación sin interrupción del servicio. Como ejemplo, recarga y reinicia una réplica defectuosa que proporciona el contexto de datos de entrada (también conocido como contexto de ejecución funcional) a partir de una réplica correcta mientras que todo el software de aplicación (ASW) (ASW #1, ASW #2 y ASW #3) no progresa. Esta función de recuperación pretende mejorar la disponibilidad de la aplicación de carga útil. Esta recuperación por redundancia tiene sus raíces en sistemas que tienen propiedades de redundancia. La característica principal realizada es la existencia de varias instancias (duplicadas como mínimo) procesando las mismas entradas y funcionando simultáneamente. Este es uno de los objetivos de la invención.
Esta estrategia de recuperación tiene que usar los datos producidos de una instancia válida para reemplazar los datos incorrectos de la instancia errónea en un punto en el tiempo. Esta técnica de reemplazo se basa en una sincronización operacional que entrega un estado estable definido entre las réplicas que funcionan en paralelo. Este es uno de los objetivos de la invención del componente de comunicación y sincronización. Detecta un comportamiento erróneo, reporta este error a la gestión de fallos para un diagnóstico y reacciona adecuadamente. En este extremo, se bloquean las operaciones de reintegración en línea (es decir, la recuperación no se realiza en segundo plano para la actividad de aplicación de carga útil): durante este intervalo de tiempo de recuperación, la aplicación de carga útil no realiza ningún progreso con el fin de mantener la coherencia del contexto de datos de

entrada. De acuerdo con una realización de la invención, el módulo gestor de recuperación se implementa dentro de la capa de software del hipervisor #4. Se hace funcionar en modo de ejecución de privilegios y se llama al final de los períodos de ronda de procesamiento mientras las réplicas están esperando.

Un componente 250 de gestión de fallos es un componente que está a cargo del diagnóstico de fallos y de la reacción a los fallos basándose en la información de detección de fallos reportada por los diferentes componentes de marco. Es responsable de ejecutar la reacción adecuada ante los fallos y manejar la gestión de redundancia: el estado de permanencia. Para lograr un alto nivel de tolerancia a fallos SEU y SEFI, la invención tiene como objetivo manejar los siguientes errores:

- "error de caída": un SEU/SEFI conduce o bien a una réplica que no envía más mensajes; o el enlace entre dos réplicas está roto o una réplica de envío está caída/bloqueada;
- "error de valor defectuoso": un SEU conduce al envío de mensajes ilegales en una réplica;
- "error bizantino": una réplica envía un mensaje diferente (pero autenticado) a las réplicas vecinas en la misma ronda;
- "error de temporizador": una réplica inicia un nuevo punto de sincronización antes de que el período mínimo de la ronda haya expirado.

La detección de estos estados se dispersa alrededor de todos los componentes funcionales TSwR FT-Fwk que son responsables del informe de fallos a través de una interfaz de gestión de fallos bien definida. La invención tiene como objetivo mejorar la disponibilidad de la aplicación de carga útil ejecutada en un procesador moderno COTS y para lo cual, la gestión de fallos gestiona el tipo de estado de fallos en lo sucesivo en el presente documento y las acciones respectivas:

- "Advertencia": el CE #x detecta un fallo emitido por un vecino tal como un mensaje faltante, mientras expira el tiempo de espera. De acuerdo con una realización de la invención, esto conduce a una información de estado sin reacción para el CE #x.
- "Error severo": el CE #x genera un fallo que detecta tal como la caducidad de un guardián de núcleo. De acuerdo con una realización de la invención, esto conduce a la gestión de redundancia con una acción para aislar esta réplica defectuosa y una solicitud de reintegración futura.
- "Error fatal": este fallo corresponde a un SEU o SEFI no recuperable que conduce a un reinicio de toda la plataforma informática (CP). Un ejemplo es un estado de bloqueo de software/hardware que desencadena el guardián del procesador o los votantes no encuentran el voto mayoritario.

Junto con el votante, las actividades de supervisión de comunicación y sincronización se gestiona un estado de permanencia de los CE (#1, #2 y #3). En el caso de que un CE falle, el protocolo de sincronización (por ejemplo, expira un tiempo de espera) o un comportamiento de un mensaje del CE se detecta defectuoso (por ejemplo, un error de contenido de mensaje), se reporta un error a la gestión de fallos. Esta capa de software diagnostica el estado y el CE erróneo se convierte en un no miembro para los votantes distribuidos y se aísla. Un CE no miembro necesita recuperarse antes de volverse miembro nuevamente.

La figura 3 ilustra un ejemplo de una secuencia de comunicación y sincronización.

1. El software de aplicación de réplica (ASW) se introduce en el punto de bloqueo síncrono que pasa a la unidad de datos de carga útil (PDU) de ronda actual;
2. El módulo de capa de sincronización sincroniza esta unidad de datos de carga útil (PDU) con una marca de tiempo global;
3. A continuación, un módulo (321, 322, 323) de capa de comunicación lo formatea (por ejemplo, encabezado, carga útil, suma de comprobación, campos de finalizador) y agrega su propio código de extensión de seguridad de réplica (también conocido como código parcial de acuerdo con el número de réplica en el que reside) al campo de finalizador;
4. El mensaje con este código de extensión de seguridad se envía de acuerdo con el procedimiento del protocolo por los medios de transmisión a otros elementos informáticos (CE) y por registro en la caja de depósito del votante (231, 232, 233) local;
5. Cada elemento informático (CE) registra en su propia caja de depósito de votantes los mensajes transmitidos (que contienen las PDU redundantes pero diferentes códigos de extensión de seguridad) y lanza el mecanismo de votación;
6. Cada módulo (231, 232, 233) de capa de votante local compara la exactitud de sus mensajes (por ejemplo, una comparación de datos byte a byte y un examen de código de seguridad) y a continuación, en el caso de acuerdo, se construye un mensaje votado de canal externo con un código de seguridad de extensión completa (etapa 300);
7. El CE #3 configurado como primario y no marcado como defectuoso (de lo contrario, es el CE #1 configurado como secundario), envía el mensaje votado al canal externo a través del medio de interfaz de E/S seleccionado (etapa 310).

El mecanismo de votación se describe a continuación.

En el ejemplo de la figura 3, se ilustran los intercambios de los mensajes relevantes-seguridad para Voto “2oo3”. Cada votante (231, 232, 233) se instancia en cada réplica (dentro de #100, #200 y #300). Cada votante asocia los mensajes correspondientes y coloca este triplete en una caja de depósito. Los mensajes de triplete en la caja de depósito se votan con un modo de votación usado para votar los mensajes con una extensión de seguridad, por ejemplo, con un procedimiento de votación “un canal seguro” (OCS). Los mensajes defectuosos se detectan y enmascaran junto con las actividades de votación y las anomalías se reportan al componente 250 de gestión de fallos. Este procedimiento de votación puede definirse para 2 de 3 votos de confirmación tardía: dos mensajes de los tres mensajes deben ser idénticos y el resultado acordado se confirma cuando se han recibido todos los mensajes o expira un tiempo de espera predefinido. El procedimiento de comparación y el procedimiento de diagnóstico (en caso de error de comparación) distinguen entre campo de datos y extensión de seguridad.

Un procedimiento o estrategia correspondiente a modo de ejemplo puede ser:

- comparar mensajes sin extensión de seguridad (por ejemplo, un AND lógico bit a bit, byte a byte o palabra a palabra). Si al menos dos mensajes no se consideran correctos hasta el momento, se reporta un error fatal;
- comparar las extensiones de seguridad de los mensajes que se consideran correctos. Si al menos dos mensajes junto con sus extensiones de seguridad no son correctos, se reporta un error fatal;
- generar un mensaje votado con la extensión de seguridad completa. Si puede construirse un mensaje votado, los datos de usuario se toman a partir de un mensaje que no se ha generado en el propio elemento informático (CE). Además, los bytes para construir la extensión de seguridad completa deben tomarse a partir de diferentes mensajes.

Se describe un enfoque de software distribuido (también conocido como prevención de errores de punto único) que implica diseñar una gestión de fallos en dos capas jerárquicas: una correlacionada con cada votante (es decir, instanciado en cada entidad de réplica) perteneciente a los CE y otra dentro del hipervisor #4 dedicado a administrar la totalidad de la plataforma informática (CP). La última, se ejecuta bajo demanda (también conocida como llamada a la interfaz de reportes) en el núcleo de llamada, lo que significa que pueden ejecutarse al mismo tiempo de 1 a 3 instancias en el modo de ejecución de privilegios más alto.

La figura 4 ilustra la línea de tiempo de un escenario a modo de ejemplo dinámico, nominal y defectuoso. Representa un ejemplo de un escenario dinámico de un TSwR FT-Fwk en un caso nominal (sin eventos SEU o SEFI) y un caso defectuoso (se produce un SEFI y detiene la ejecución de réplica CE #3). El escenario a modo de ejemplo se divide en varias fases identificadas con referencias numeradas.

Se operan tres CE (CE #1, CE #2 y CE #3) a partir de segmentos de memoria separados en diferentes núcleos de procesamiento. La probabilidad de que un evento SEU o SEFI bloquee el procesador se reduce de este modo. Los componentes funcionales de la arquitectura TSwR FT-Fwk instanciados en cada elemento informático (CE) detectan y enmascaran el evento SEU que impacta en una localización de memoria de datos. También detecta el evento SEFI en una réplica y esta réplica defectuosa se aísla y a continuación se recupera.

En el ejemplo, el CE #3 es defectuoso. Los procedimientos y sistemas descritos pueden recargar completamente el CE erróneo (por ejemplo, una configuración de hardware virtual, un software de aplicación (ASW) y programas FT-Fwk) y a continuación reiniciar la fase de inicialización durante la que el contexto de datos de entrada se sobrescribe con los correctos provenientes de un CE válido (lo que implica que uno existe).

La fase 1 (etapa 401) corresponde a la puesta en marcha y la inicialización de la totalidad de la plataforma informática (CP). El escenario comienza a partir del encendido de la plataforma informática (CP) (tiempo marcado t0) con la inicialización completa de la plataforma informática (CP). La capa de hipervisor #4 carga e inicia todos los elementos informáticos (CE) usando solo un programa almacenado en una memoria no volátil. De acuerdo con la configuración de las máquinas virtuales predefinidas apropiadamente, el hipervisor #4 carga tres instancias de este programa en tres segmentos de RAM aislados.

El ti, ReadyToStart es un punto en el tiempo cuando todas las réplicas (#1, #2 y #3) llaman a la gestión de fallos distribuidas para el diagnóstico y las reacciones con respecto a su estado.

La fase 2 (etapa 402) corresponde a las etapas de gestión de fallos que incluyen el diagnóstico de estado de plataforma informática (CP) y las reacciones respectivas. En este ejemplo, las réplicas entran en esta fase con un comportamiento correcto que da como resultado una reacción nominal; todas las réplicas son una permanencia de los votantes y no se realiza ninguna acción de recuperación. El ti, SynchroStart es un punto en el tiempo cuando todas las réplicas están bloqueadas esperando una señal de sincronización para comenzar una nueva ronda de procesamiento. La duración del tiempo (ti, SynchroStart - ti, ReadyToStart) es el intervalo de tiempo para la gestión de fallos distribuidos y la estrategia de recuperación. Un intervalo de tiempo de este tipo puede dominarse y limitarse ventajosamente. Durante este punto de bloqueo sincronizado, el tiempo global, el número de ronda y el contexto de datos de entrada se actualizan para cada réplica.

La fase 3 (etapa 403) corresponde al intervalo de tiempo de ejecución de carga útil de aplicación. La corrección de comportamiento de la ronda de procesamiento del software de aplicación (ASW) se monitoriza con una ventana de

tiempo (también conocido como permitir una fluctuación de tiempo de ejecución) alrededor del ti predefinido, SynchroPoint con un retardo mínimo *dlmín* y un retardo máximo *dlmáx* para alcanzar este punto. El ti, SynchroPoint es un punto en el tiempo cuando todos los software de aplicación (ASW) (ASW #10, ASW #20 y ASW #30) han producido valores de datos de salida y han llamado a la interfaz de tiempo de ejecución de aplicación para construir un mensaje PDU para la comunicación y la sincronización. Desde el ti, SynchroPoint hasta ti + 1, SynchroStart, se suspende la carga útil de aplicación. El período de tiempo de ronda de procesamiento [*Tp = ti + 1, SynchroStart - ti, SynchroStart]* puede dominarse y limitarse ventajosamente. Esta duración se monitoriza y la falta de cumplimiento es un error no recuperable.

La fase 4 (etapa 404) corresponde a la globalización, el voto distribuido y el intervalo de tiempo de comunicación exterior. La capa de comunicación y sincronización de cada FT-Fwk (#100, 200 y #300) realiza las etapas del protocolo de seguridad para enviar (de forma punto a punto) los propios mensajes a las réplicas vecinas. A continuación, el procedimiento de votación se desarrolla con actividades de comparación, acuerdo y compromiso. Esta fase finaliza bien y, de acuerdo con el ejemplo, el elemento informático CE #3, propietario del canal principal externo, transmite al exterior su mensaje votado (incluida la extensión de seguridad completa) antes del fin de plazo *ti + 1*, *ReadyToStart*. El *ti + 1*, *ReadyToStart* es un punto de sincronización en el tiempo cuando las réplicas (CE #1, CE #2 y CE #3) llaman a la gestión de fallos distribuidos para el diagnóstico y las reacciones con respecto a su estado.

La fase 5 (etapa 405) corresponde a las actividades o etapas de gestión de fallos que incluyen también el diagnóstico del estado de plataforma informática (CP) y sus reacciones respectivamente. En el ejemplo, FT-Fwk #100 y #200 llegan aproximadamente al mismo tiempo sin que se detecte ningún fallo, el FT-Fwk #300 entra en esta etapa después y también sin detectar ningún fallo. Por lo tanto, este comportamiento correcto da como resultado una reacción nominal; todas las réplicas mantienen la permanencia de los votantes y no se necesita una estrategia de recuperación.

La fase 6 (etapa 406) comienza con *ti + 1*, el punto de sincronización SynchroStart cuando toda la carga útil de las aplicaciones (ASW #10, ASW #20 y ASW #30) se suspende a la espera de una señal de sincronización para iniciar una nueva ronda de procesamiento. La fase anterior resume cada réplica con el nuevo tiempo global calculado, el número de la próxima ronda y el contexto de datos de entrada grabados. En el ejemplo, hay un impacto de partículas (radiación) en el procesador físico y el escenario a modo de ejemplo supone que esta partícula produce un evento SEFI en el núcleo de procesamiento asignado al elemento informático CE #3. El efecto del impacto corresponde a un estado de bloqueo del núcleo que podría dar como resultado una situación de bucle sin fin, una excepción de comprobación de máquina u otra situación de bloqueo en función de la arquitectura de procesador seleccionada. Durante el intervalo de tiempo de ejecución de carga útil de aplicación, la ejecución ASW #30 del software de aplicación (ASW) se corrompe y el núcleo asociado entra en un estado de bloqueo que se detectará por uno de los mecanismos de monitorización de estado (por ejemplo, un guardián, un controlador de excepción, un tiempo de espera de fin de plazo). En algunas realizaciones, el componente de monitorización de estado reporta una información de estado de error severo al componente de gestión de fallos que conduce a la gestión de redundancia con una acción para aislar esta réplica. Mientras tanto, los softwares de aplicación (ASW) ASW #10 y ASW #20 continúan sus rondas de procesamiento para producir valores de datos de salida y a continuación llaman a la interfaz de tiempo de ejecución de aplicación que entra en *ti + 1*, el punto de sincronización *SynchroPoint*. En una realización, cuando se desencadena un tiempo de espera *dlmáx* (por ejemplo debido al software de aplicación faltante (ASW) #30), tanto el FT-Fwk #100 como el #200 reportan una información de estado de advertencia al componente de gestión de fallos que a su vez confirma la acción de gestión de redundancia: el elemento informático CE #3 se aísla, a continuación, de los votantes. La fase 6 despliega la globalización del mensaje, el voto distribuido y las etapas de comunicación externa como se han introducido en la fase 4 (etapa 404). Los restantes FT-Fwk #100 y #200 se envían mensajes propios entre ellos mismos. A continuación, el procedimiento de votación se aplica solo a los dos miembros restantes. En el ejemplo, la fase termina con el elemento informático CE #1; el propietario secundario de canal externo y la transmisión de su mensaje votado al exterior. La etapa de recuperación se describe a continuación. En el punto de sincronización *ti + 2*, *ReadyToStart*, las dos réplicas restantes (#1 y #2) llaman al componente de gestión de fallos distribuidos que pasa a la gestión de fallos en capas de hipervisor #4.

La fase 7 (etapa 407) se realiza principalmente por la capa de hipervisor #4 que contiene la gestión de fallos de plataforma informática (CP) y el componente de estrategia de recuperación. La gestión de fallos de plataforma informática (CP) diagnostica un error severo del CE #3 y a continuación activa la estrategia de recuperación. La recuperación puede consistir, por ejemplo, en que el hipervisor #4 cargue el programa en un segmento de memoria RAM predefinido y comience el programa desde el principio. Este elemento informático CE #3 reiniciado (re)une las otras réplicas en el punto de sincronización *ti + 2*, SynchroStart.

REIVINDICACIONES

1. Un procedimiento implementado por ordenador para detectar un fallo en un sistema que comprende las etapas de:

ejecutar al menos tres máquinas (1, 2, 3) virtuales, ejecutando cada máquina virtual un mismo software (10) de aplicación, en segmentos (1.2, 2.2, 3.2) de memoria separados y aislados, y en un núcleo (1.1, 2.1, 3.1) dedicado de un procesador multinúcleo;
estando dichas máquinas (1,2,3) virtuales sincronizadas y ejecutándose simultaneamente por un hipervisor (4) común; en el que unas máquinas virtuales no defectuosas proporcionan un mensaje de salida idéntico dentro de un intervalo de tiempo predefinido;
detectar un fallo (250) en una salida de una máquina virtual, correspondiendo dicho fallo a un mensaje de salida diferente de dicha máquina virtual defectuosa;
ejecutar un voto (230) distribuido en los mensajes de salida de las máquinas virtuales para determinar un mensaje (400) de salida votado;
comunicar el mensaje (400) de salida votado a un sistema externo en comunicación con el sistema;
en el que una máquina virtual está predefinida como primaria y otra como secundaria, y en el que el mensaje (400) de salida votado se comunica por la máquina virtual primaria o por la máquina virtual secundaria si la máquina virtual primaria está defectuosa.

2. El procedimiento de la reivindicación 1, en el que cada máquina virtual extrae o inserta un mensaje de salida en las otras máquinas virtuales.

3. El procedimiento de la reivindicación 2, en el que los mensajes de salida de cada máquina virtual se recopilan en una caja de depósito y se determina un mensaje de salida votado a partir de los mensajes de salida.

4. El procedimiento de cualquier reivindicación anterior, en el que los mensajes de salida de las máquinas virtuales están numerados y/o están marcados en el tiempo y/o estructurados y/o anotados.

5. El procedimiento de cualquier reivindicación anterior que comprende además la etapa de asociar una extensión de seguridad con un mensaje de salida de una máquina virtual, comprendiendo dicha extensión de seguridad una información de identificación acerca de la máquina virtual que emite el mensaje de salida.

6. El procedimiento de cualquier reivindicación anterior, que comprende además la etapa de recuperar la máquina virtual defectuosa.

7. El procedimiento de la reivindicación 6, en el que la etapa de recuperar la máquina virtual defectuosa comprende usar el contexto de datos de una máquina virtual no defectuosa para reemplazar el contexto de datos de la máquina virtual defectuosa.

8. El procedimiento de las reivindicaciones 6 o 7, en el que la etapa de recuperación de la máquina virtual defectuosa se realiza en un punto de resincronización en el tiempo.

9. El procedimiento de cualquier reivindicación anterior, en el que un fallo está asociado con un error elegido de la lista que comprende: error de caída, error de valor defectuoso, error bizantino, error de temporizador y combinaciones de los mismos.

10. Un programa informático que comprende instrucciones para realizar las etapas del procedimiento de acuerdo con una cualquiera de las reivindicaciones 1 a 9 cuando dicho programa informático se ejecuta en un dispositivo informático adecuado.

11. Un medio legible por ordenador que tiene codificado en el mismo un programa informático de acuerdo con la reivindicación 10.

12. Un sistema que comprende unos medios adaptados para realizar las etapas del procedimiento de acuerdo con una cualquiera de las reivindicaciones 1 a 9.

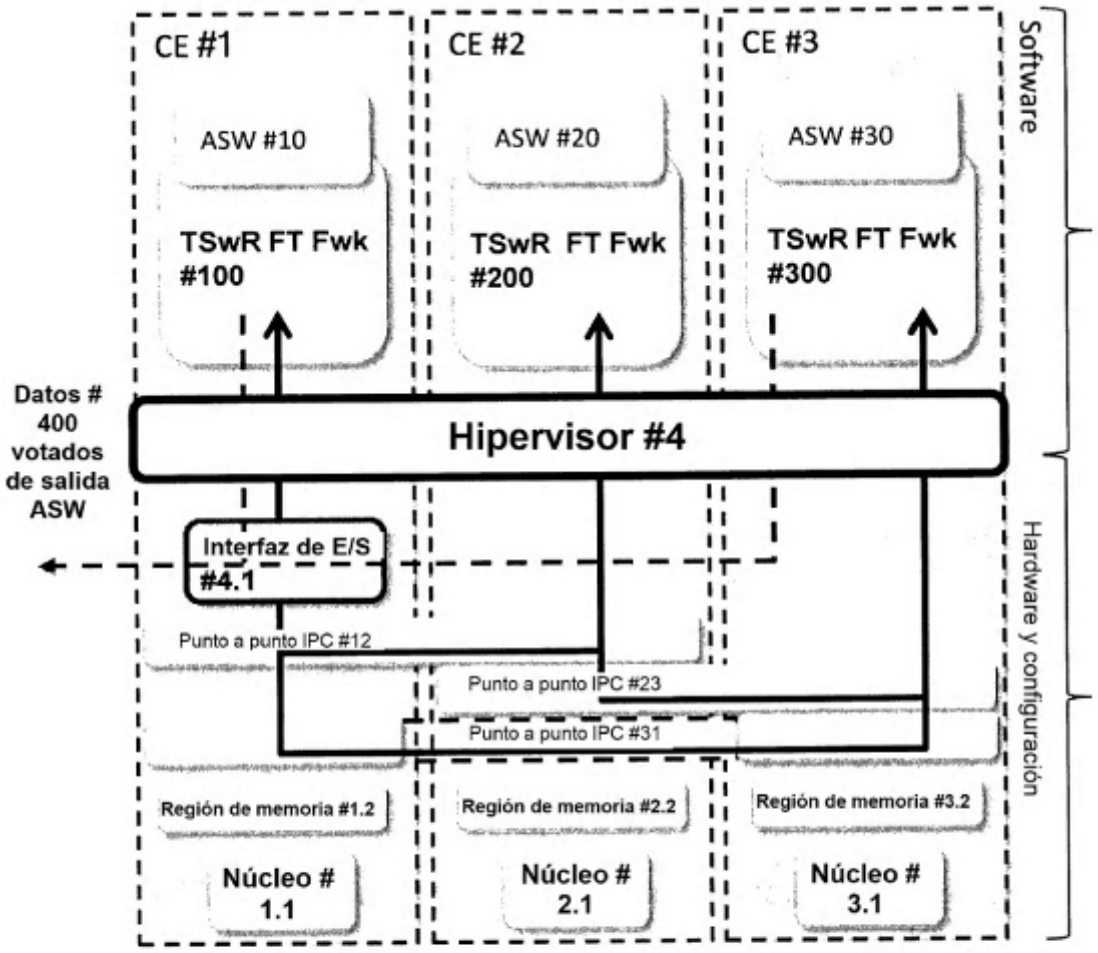
CE #1
CE #2
CE #3
Software
ASW #10
ASW #20
ASW #30
TSwR FT Fwk #100
TSwR FT Fwk #200
TSwR FT Fwk #300
Datos # 400 votados de salida ASW
Hipervisor #4
Interfaz de E/S #4.1
Punto a punto IPC #12
Punto a punto IPC #23
Punto a punto IPC #31
Región de memoria #1.2
Región de memoria #2.2
Región de memoria #3.2
Núcleo # 1.1
Núcleo # 2.1
Núcleo # 3.1
Hardware y configuración

FIG.1

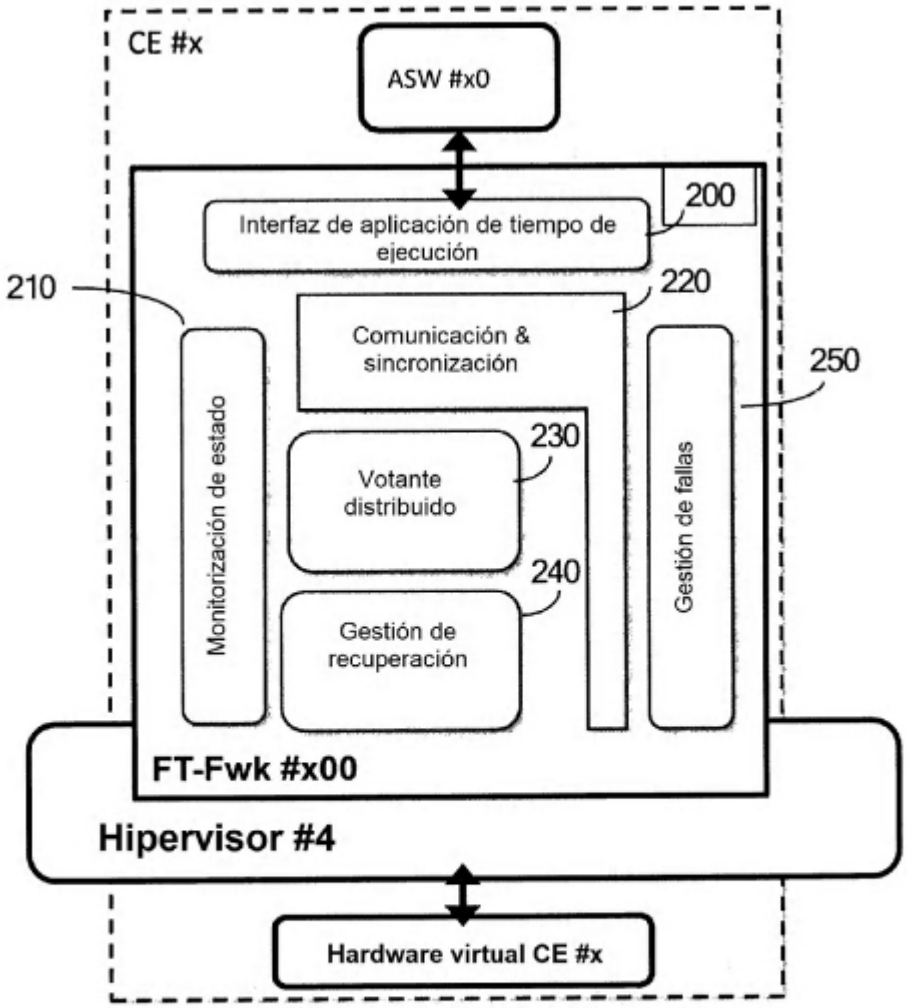
CE #x
ASW #x0
200
Interfaz de aplicación de tiempo de ejecución
210
220
Comunicación & sincronización
250
Monitorización de estado
230
Votante distribuido
240
Gestión de recuperación
Gestión de fallas
FT-Fwk #x00
Hipervisor #4
Hardware virtual CE #x

FIG.2

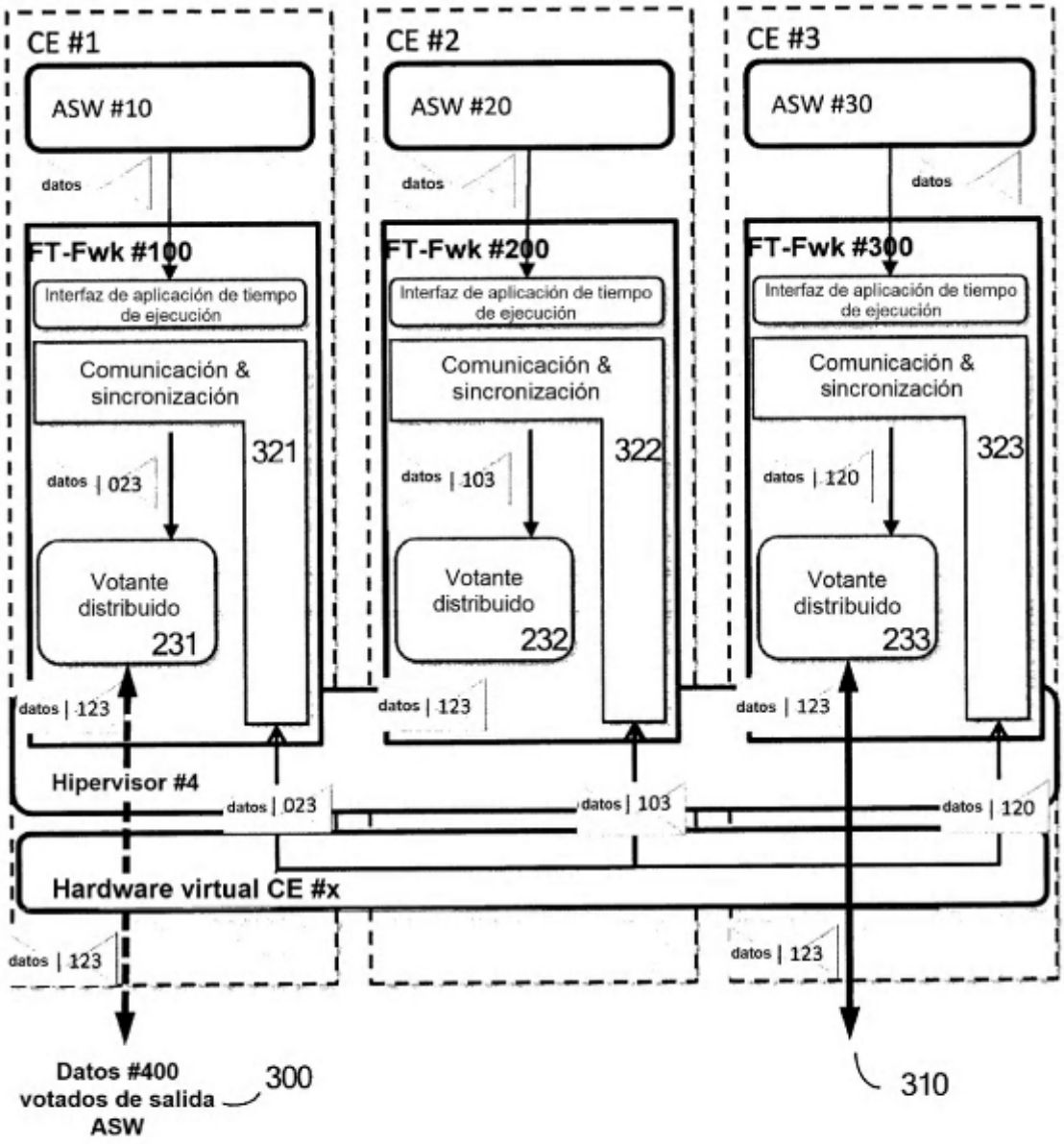
CE #1
ASW #10
datos
FT-Fwk #100
Interfaz de aplicación de tiempo de ejecución
Comunicación & sincronización
321
datos | 023
Votante distribuido
231
datos | 123
CE #2
ASW #20
datos
FT-Fwk #200
Interfaz de aplicación de tiempo de ejecución
Comunicación & sincronización
322
datos | 103
Votante distribuido
232
datos | 123
CE #3
ASW #30
datos
FT-Fwk #300
Interfaz de aplicación de tiempo de ejecución
Comunicación & sincronización
323
datos | 120
Votante distribuido
233
datos | 123
Hipervisor #4
datos | 023
datos | 103
datos | 120
Hardware virtual CE #x
datos | 123
datos | 123
Datos #400
votados de salida
ASW
300
310

FIG.3

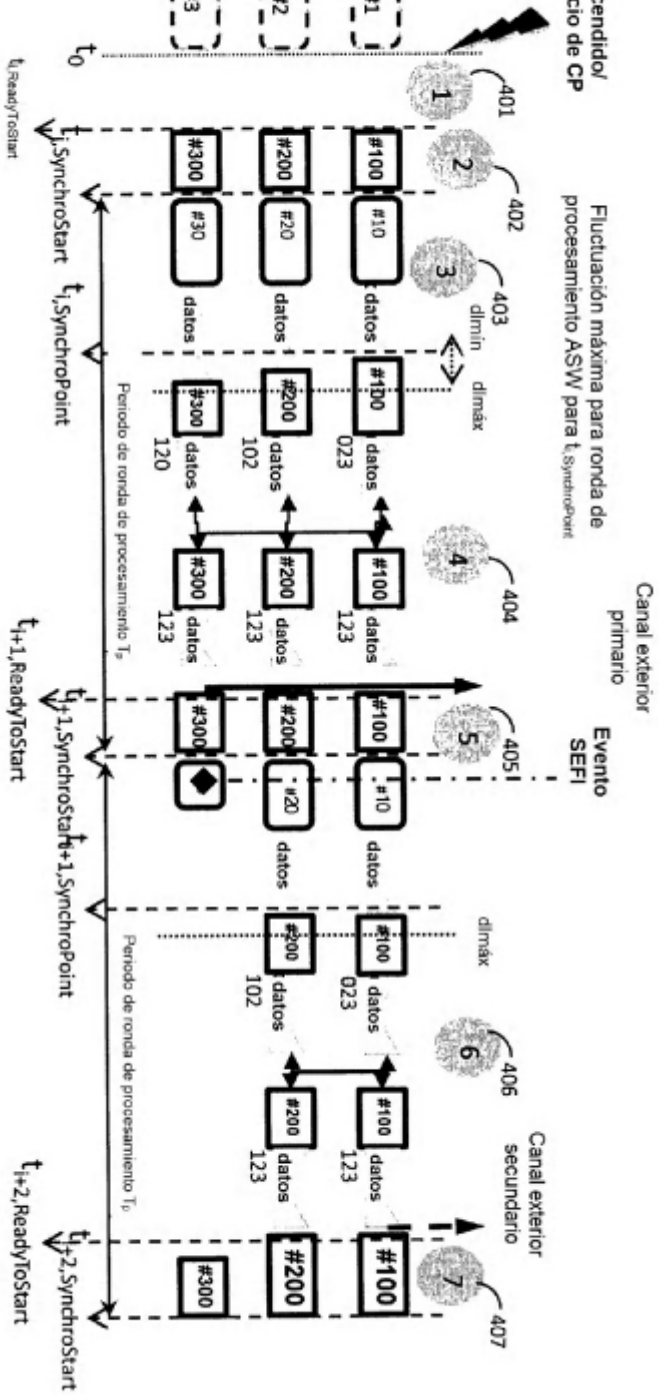
Encendido/ reinicio de CP
Fluctuación máxima para ronda de procesamiento ASW para t,SynchroPoint
Canal exterior primario
Evento SEFI
Canal exterior secundario
CE #1
CE #2
CE #3
401
402
403
404
405
406
407
dlmín
dlmáx
datos
Periodo de ronda de procesamiento Tp
t0
t,ReadyToStart
ti,SynchroStart
ti,SynchroPoint
ti+1,ReadyToStart
ti+1,SynchroStart
ti+1,SynchroPoint
ti+2,ReadyToStart
ti+2,SynchroStart

FIG.4