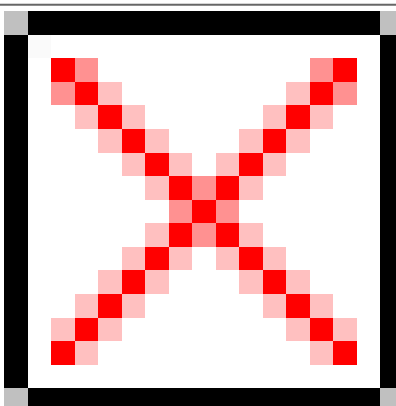


DIRECCIÓN DE NUEVAS CREACIONES

SOLICITUD FASE NACIONAL - PCT



1	Título de la Invención (200 caracteres o espacios máximos)		
FALLO DEL CLIENTE EN CLÚSTER			
2	Datos del Solicitante / Titular		
Nombre:	MICROSOFT CORPORATION	Dirección Electrónica:	infopi@cardenasycardenas.com
Dirección:	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399	Domicilio/País de constitución:	ESTADOS UNIDOS DE AMÉRICA - WASHINGTON - REDMOND
Identificación:			
☐ CEDULA DE CIUDADANIA ☒ EMPRESA EXTRANJERA ☐ PASAPORTE		☐ CEDULA DE EXTRANJERIA ☐ NIT	

Número:		1965228722-	
3	Solicitantes		
Apellidos - Nombres o Razón Social		Tipo	Identificación
1.	MICROSOFT CORPORATION	EE	1965228722
4	Datos del Inventor		
Nombre:		Thomas, E. JOLLY	Dirección Electrónica: infopi@cardenasycardenas.com
Dirección:		c/o Microsoft Corporation, LCA-International Patents, One Microsoft Way, 98052-6399	Domicilio/País de constitución: ESTADOS UNIDOS DE AMÉRICA - WASHINGTON - REDMOND
Identificación:			
☐ CEDULA DE CIUDADANIA ☐ EMPRESA EXTRANJERA ☐ PASAPORTE ☐ CEDULA DE EXTRANJERIA ☐ NIT ☒ No Aplica			
Número:		-	
5	Inventor(es)		
Apellidos - Nombres		Domicilio	
1.	KRUSE David, M	ESTADOS UNIDOS DE AMÉRICA	
2.	FATHALLA Diaa	ESTADOS UNIDOS DE AMÉRICA	
3.	PINKERTON James, T	ESTADOS UNIDOS DE AMÉRICA	
4.	GEORGE Mathew	ESTADOS UNIDOS DE AMÉRICA	
5.	PRAHALAD Prashanth	ESTADOS UNIDOS DE AMÉRICA	
6.	JOLLY Thomas, E.	ESTADOS UNIDOS DE AMÉRICA	
6	Datos Inventor(es)		

País de Residencia		Departamento/Estado	Ciudad	Dirección
1.	ESTADOS UNIDOS DE AMÉRICA	WASHINGTON	REDMOND	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399
2.	ESTADOS UNIDOS DE AMÉRICA	WASHINGTON	REDMOND	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399
3.	ESTADOS UNIDOS DE AMÉRICA	WASHINGTON	REDMOND	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399
4.	ESTADOS UNIDOS DE AMÉRICA	WASHINGTON	REDMOND	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399
5.	ESTADOS UNIDOS DE AMÉRICA	WASHINGTON	REDMOND	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399
6.	ESTADOS UNIDOS DE AMÉRICA	WASHINGTON	REDMOND	c/o Microsoft Corporation, LCA- International Patents, One Microsoft Way, 98052-6399
7	Datos del Representante Legal / Apoderado			
Nombre:		LUZ HELENA ADARVE GOMEZ	Dirección Electrónica:	infopi@cardenasycardenas.com
Dirección:		Carrera 7 No 71-52 T B Piso 9	Domicilio/País de constitución:	COLOMBIA - BOGOTA D.C. - BOGOTA D.C.
Identificación:				

☒ CEDULA DE CIUDADANIA
☐ EMPRESA EXTRANJERA
☐ PASAPORTE

☐ CEDULA DE EXTRANJERIA
☐ NIT

Número:

41575435-

Presentación de Poder

Año de Radicación

Número de Radicación

8

Datos Solicitud: PCT / WO

Número Solicitud:

pct/us2012/054038

Fecha Solicitud:

07/09/2012

Número Publicacion:

wo 2013/036697

Fecha Publicacion:

14/03/2013

9

Declaraciones de prioridad

☒ SI ☐ NO

	(33) País de origen	Codigo del país	(31) No. Solicitud	(32) Fecha
1.	ESTADOS UNIDOS DE AMÉRICA	US	13/228,732	09/09/2011

10

Reivindicaciones

Número reivindicaciones:

10

Pago Reivindicaciones:

No

11

Reducción de tasas.

Declaro que carezco de medios económicos para presentar la solicitud de patente.

☐ SI ☒ NO

Nota: En caso de ser persona natural y carecer de medios económicos, y por lo tanto, aplique la reducción de tasas a que se refiere la resolución vigente en tarifas, debe firmar la presente solicitud bajo la gravedad de juramento.

- ☐ Micro, pequeñas y medianas empresas
☐ Universidades públicas o privadas
☐ Entidades sin ánimo de lucro

Debe aportar los documentos que se indican en el numeral 17 de anexos

12

Documentos Anexos

- ☒ Reivindicaciones
☒ Descripción
☒ Dibujos y/o Figuras
☒ Resumen
☐ Certificado Depósito Material Biológico
☐ Uso de Conocimiento tradicional

- ☐ Listado de secuencias
- ☒ Artes finales 12 x 12 cm
- ☒ Poderes, si fuere el caso
- ☐ Copia de la primera solicitud si se reivindica prioridad
- ☐ Traducción simple de la primera solicitud, si se reivindica prioridad
- ☒ Otros Anexos



- (51) **International Patent Classification:**
G06F 15/16 (2006.01)
- (21) **International Application Number:**
PCT/US2012/054038
- (22) **International Filing Date:**
7 September 2012 (07.09.2012)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
13/228,732 9 September 2011 (09.09.2011) US
- (71) **Applicant (for all designated States except US):** **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).
- (72) **Inventors:** **KRUSE, David, M.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **FATHALLA, Diaa**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **PINKERTON, James, T.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **GEORGE, Mathew**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **PRAHALAD, Prashanth**; c/o Mi-

crosoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **JOLLY, Thomas, E.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

- (81) **Designated States** (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) **Title:** CLUSTERED CLIENT FAILOVER

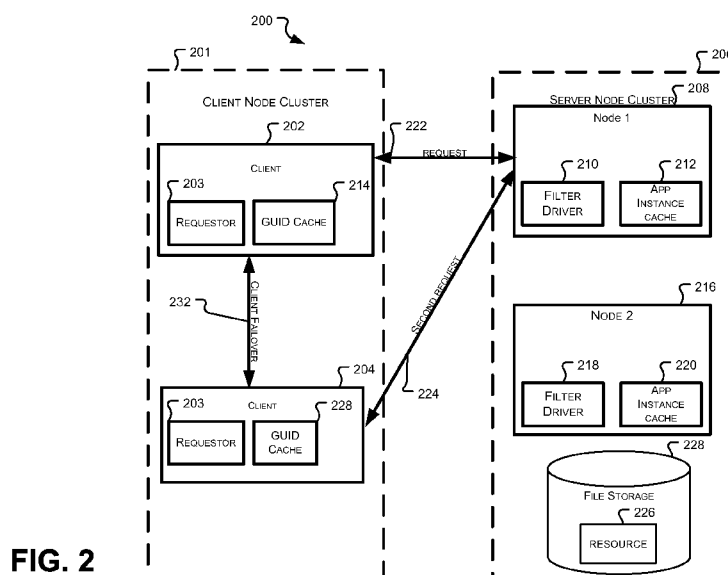


FIG. 2

(57) **Abstract:** An application instance identifier is employed with various systems and methods in order to provide a requestor with continuous access to a resource when operating in a client clustered environment. A requestor residing on a first client in may attempt to access a resource. The first client sends a request to access the resource. The request may be associated with an application instance identifier that identifies the requestor. At some point, the first client fails and the requestor is associated with a second client via a failover mechanism. The second client sends a second request to access the resource on behalf of the requestor. The second request is associated with the requestor's application instance identifier. The application instance identifier is used to identify the second request as belonging to the same requestor as the first request, thereby granting the second request to access the resource while avoiding a conflict situation.

**Declarations under Rule 4.17:**

- *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
- *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

CLUSTERED CLIENT FAILOVER

Background

[0001] Clustered environments, e.g., environments where workload is distributed across multiple machines, are commonly used to provide failover and high availability of information to clients. Clustered environments allow clients to access resources via the one or more nodes that are a part of the environment. A clustered environment can act as a client, a server, or both. In a client cluster server, an application may reside on any of the nodes that make up the cluster. The application may issue requests for resources that are stored locally within the client cluster or stored remotely. If an error occurs on the node, the client failover, or migrate, to a different node in the cluster. However, when the client again requests to access a resource that it was working with at the time of the error, the resource may be fenced or locked by the server for the previous client node that the application resided on.

[0002] It is with respect to these and other considerations that embodiments have been made. Also, although relatively specific problems have been discussed, it should be understood that the embodiments should not be limited to solving the specific problems identified in the background.

Summary

[0003] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detail Description section. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0004] Systems and methods are disclosed herein that provide an application or a process with continuous access to a resource after the application migrates to a new node in a clustered client environment. An application or process residing on a node in a client cluster sends a request to a server to access a resource. In embodiments, a unique application instance identifier is used to identify an application requesting a resource. The unique application identifier may be provided with the request. When the client accesses a resource, the application instance identifier is associated with the requested resource.

[0005] Before the application or process completes its operations on the resource, the node upon which the client resides in the clustered environment may experience an error that causes it to fail or otherwise lose access to the resource prior to the application properly releasing the resource. In such circumstances, the resource may remain in a

fenced or locked state on the server per the previous client's request. Upon failing over to a different node in the client cluster, the application on the new client node may reestablish a connection with the server managing the resource and make a second request for the resource that the application previously had access to at the time of the error. The second request may include the application instance identifier was sent with the first request. Although the second request for the resource may be received from a different node in the clustered environment, the application instance identifier permits the server managing the request to determine that the second request belongs to the same application or process that had previously locked the resource. Doing so allows the server to invalidate the resource and grant the client's second request to access the resource while insuring a conflict situation does not arise.

[0006] Embodiments may be implemented as a computer process, a computing system or as an article of manufacture such as a computer program product or computer readable media. The computer program product may be a computer storage media readable by a computer system and encoding a computer program of instructions for executing a computer process. The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process.

Brief Description of the Drawings

[0007] Non-limiting and non-exhaustive embodiments are described with reference to the following figures.

[0008] FIG. 1 illustrates a system that may be used to implement embodiments described herein.

[0009] FIG. 2 is a block diagram illustrating a software environment that may be used to implement the embodiments disclosed herein.

[0010] FIG. 3 is an embodiment of a method that a client may perform to gain continuous access to a resource in a clustered environment.

[0011] FIG. 4 is an embodiment of a method performed by a node in a clustered environment to provide continuous access to a resource.

[0012] FIG. 5 illustrates a block diagram of a computing environment suitable for implementing embodiments.

Detailed Description

[0013] Various embodiments are described more fully below with reference to the accompanying drawings, which form a part hereof, and which show specific exemplary

embodiments. However, embodiments may be implemented in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the embodiments to those skilled in the art. Embodiments may be practiced as methods, systems or devices. Accordingly, embodiments may take the form of a hardware implementation, an entirely software implementation or an implementation combining software and hardware aspects. The following detailed description is, therefore, not to be taken in a limiting sense.

[0014] Embodiments of the present disclosure are related to providing clustered client failover mechanisms that allow a requestor to regain access to a resource after a failover event. In embodiments, a requestor may be a process, an application, or one or more child processes of an application. A resource may be a file, an object, data, or any other type of resource in a computing environment. In embodiments, a resource may reside on a standalone server or it may reside in a clustered environment. In the embodiments disclosed herein, a clustered environment may include one or more nodes (e.g., client and/or server devices).

[0015] In an example embodiment, an application residing on a node in a clustered environment may request access to a particular resource. In embodiments, the resource may be stored locally (e.g., on the client node), in a remote device (e.g., a remote server or a different node in the client clustered environment), or in a clustered environment (e.g., an environment containing multiple nodes) that is different from the client clustered environment. For example, in embodiments the clustered environment may be a client or server cluster; however, one of skill in the art will appreciate that the systems and methods disclosed herein may be employed in any other type of environment, such as, but not limited to, a virtual network.

[0016] In such environments, resources may be shared among clients and applications. When an application accesses a resource, the resource may be fenced or locked, thereby prohibiting other applications from accessing the resource until the accessing application releases the resource. Fencing or locking the resource may be employed to protect against a conflict, that is, protect against modification of the resource by another application before the accessing application has performed its operations on the resource. However, if the node in a clustered client environment fails, the application accessing the resource may not properly release the resource from a fenced or locked state. For example, the client node accessing the resource on behalf of the application may lose a network connection,

may crash, or may otherwise lose access to the resource prior to the application completing its operations and properly releasing the resource. Thus, the resource may remain in a state in which it is unavailable to other clients or applications. Mechanisms may be employed that automatically release a resource from a fenced or locked state, thereby preventing the resource from being permanently locked out. However, such mechanisms often wait a period of time before releasing a fenced or locked resource.

[0017] In some instances, when the application performs a failover to migrate from the failed client node to a different client node in the client cluster, the application may attempt to reestablish its previous connection with the server and resume its operation(s) on the resource via the different client node. However, because the resource was not properly released by the failed client node, which previously accessed the resource on the application's behalf, due to the error, the application that had previous access to the resource may not be able to resume its access of the resource until the server releases the resource from its fenced or locked state. However, because a different node is now attempting to access the resource on the application's behalf, the server may not be able to identify the application as the same application that previously established the lock on the resource. However, because the same application is trying to access the resource, a conflict situation does not exist. In such situations, waiting for the server to release the previous lock on the resource may cause an unacceptable delay for the application.

[0018] As described, because the application is operating in a clustered client environment, when the application requests to access the resource a second time, the request to access to the resource may be made from a different location, such as, a different node in the client clustered environment. Thus, the second request may come from a location or different IP address. Because the request may be made from a different location, a server may have difficulty ensuring that the client or application attempting to again access the resource is actually the same client that previously accessed the resource. The systems and methods disclosed herein provide mechanisms to identify situations where the same application is attempting to access a resource, thereby avoiding such delay and providing an application continuous access to the resource.

[0019] FIG. 1 illustrates a system 100 that may be used to implement some of the embodiments disclosed herein. System 100 includes client cluster 102 and a server cluster 106. Client cluster includes multiple nodes, such as clients 102A and 102B. Clients 102A and 102B may be a device or application residing in client cluster 102. Client cluster 102 may communicate with server cluster 106 through network 108. In embodiments, network

108 may be the Internet, a WAN, a LAN, or any other type of network known to the art. Server cluster 106 stores resources that are accessed by applications on client cluster 102 (e.g., applications residing on Client 102A or Client 102B). In embodiments, a client (e.g., Client 102A) may establish a session with cluster 106 to access the resources on
5 cluster 106 on behalf of an application residing on the client. Although in FIG. 1 client cluster 102 only includes two clients (e.g., Client 102A and Client 102B), one of skill in the art will appreciate that any number of clients may be included in client cluster 102.

[0020] As shown in FIG. 1 server cluster 106 includes servers 106A, 106B, and 106C, which provide both high availability and redundancy for the information stored on cluster
10 106. In embodiments, the cluster 106 may have a file system, a database, or other information that is accessed by clients 102 and 104. Although three servers are shown in FIG. 1, in other embodiments cluster 106 may include more than three servers, or fewer than three servers. Furthermore, while the embodiments herein described relate to a client communicating with a server that is part of a server cluster, one of skill in the art will
15 appreciate that the embodiments disclosed herein may also be performed using a standalone server.

[0021] In embodiments, client cluster 102 provides failover mechanisms that allow a client to migrate from a first client node to a second client node in case of an error or failure occurring on the first client node. One of skill in the art will appreciate that any
20 type of failover mechanism may be employed with the systems and methods disclosed herein. The methods and systems disclosed herein may be employed to avoid undue delay when an application attempts to regain access to a resource migrating from one client to another (e.g., from Client 102A to Client 102B) in the case of a failover. In embodiments, an application instance identifier identifying the application accessing the resource may be
25 associated with the resource. The application instance identifier may be a globally unique identifier (GUID) that is associated with an application, an action performed by an application, or a child process of an application. For example, in one embodiment an application may be associated with an application instance identifier that is a GUID. In another embodiment, an application instance identifier may be associated with a specific
30 operation or action performed by an application. For example, if the application issues two different open requests for two different files, each open request may have its own application instance identifier. In yet another embodiment, an application instance identifier may be associated with one or more child processes of the application. As will be clear to one of skill in the art from the embodiments described herein, associating the

application instance identifier of an application with its one or more child processes will allow the child processes to access the resource if the resource is placed in a locked or fenced state that belongs to the application. In embodiments, the application instance identifier may be sent by the client when, or after, sending a request for a resource.

5 **[0022]** In accordance with another embodiment, in addition to storing information accessed by clients that are a part of client cluster 102, server cluster 106 also provide a failover mechanism that allows continuous access of a resource in case of a server node failure. Again, one of skill in the art will appreciate that any type of failover mechanism may be employed with the systems and methods disclosed herein.

10 **[0023]** In embodiments, when a client requests access to a resource on behalf of an application, the application instance identifier of the application is sent with the request. The server receiving the request may associate the application instance identifier with the resource. For example, the server cluster may store the application instance identifier in a table or cache located on one or more nodes (e.g., servers such as servers 106A, 106B,
15 and/or 106C) located in the server cluster 106 in such a manner that the application instance identifier is associated with the resource. Before the client is done with the resource, the client may experience an error that will force it to lose connection with the resource. For example, the client hosting the application or performing requests or operations on the application's behalf may lose its network connection to the server
20 cluster, the client may crash, or any other type of error may occur that interferes with the applications use of the resource. Upon experiencing the error, the application may failover to a new client node in the client cluster 102. The new client node may reconnect to the server cluster and send a second request to access the resource on the application's behalf. In embodiments, the client may reconnect to the same node in the sever cluster 106 or a
25 different node. The second request to access the resource may include the application instance identifier of the application. Upon receiving the second request, the sever (e.g., a Server 106A of server cluster 106) compares the application instance identifier of the second request with the application instance identifier associated with the resource. If the two application instance identifiers match, the server cluster invalidates the resource. In
30 embodiments, invalidating the resource may comprise closing a file, removing a lock on the resource, or otherwise taking any action that frees the resource for use. The server node may then grant the application's second request to access the resource. If the application instance identifier of the second node does not match application identifier

associated with the resource, the server will not allow access to the resource until the resource becomes free.

[0024] To illustrate one embodiment, a requestor (e.g., a process, application, etc.) on Client 102A in client cluster 106 may request that Client 102A establishes a session with a server of server cluster 106. For example, Client 102A may establish a session with server 106A to access a database stored that is on server 106A or that is a part of server cluster 106, in which server 106A may access the database. Client 102A then sends a request for a resource on behalf of the requestor. An application instance identifier that identifies the requestor is associated with the request. In embodiments, the request may include the application instance identifier or the application instance identifier may be sent separately in a manner such that the server 106A can determine that the application instance identifier is associated with the request. In yet another embodiment, the server 106A or the server cluster 106A may already have information needed to associate the application instance identifier with the request without having to receive the application instance identifier along with the request. Server 106A then grants the requestor access to the resource, thereby allowing the requestor to perform operations on or otherwise access the resource. When granting the requestor access to the resource, server 106A associates an application instance identifier with the resource in a manner that indicates the requestor is currently accessing the resource. The resource may then be fenced or locked so other clients or applications cannot access or modify the resource until client 102 has completed its operation.

[0025] Before the requestor completes its operations on the resource, an error occurs that causes Client 102A to fail or to otherwise lose its connection to the resource. Because client requestor completed its operation, it has not released control of the resource. Thus, the resource may remain in a fenced or locked state. The requestor or client cluster 102 may employ a failover mechanism to migrate the requestor from client 102A to client 102B. Once the failover operation is complete, client 102B may reconnect to server cluster 106 on the requestor's behalf. Client 102B may reconnect to server 106A or establish a new connection with any other server in server cluster 106 (e.g., server 106B or 106C). In an example situation, Client 102B reconnects to server 106A. Upon reconnecting, the client 102B may send a second request to access the resource on behalf of the requestor. As previously noted, because the requestor did not release control the resource, the resource may still be in a locked or fenced state. In order to access the resource, without waiting for the server to automatically change the state of the resource,

for example, through a time out operation, the requestor may again provide its application instance identifier with the second request. Server 106A compares the application instance identifier provided with the second request to the application instance identifier associated with the resource. For example, by comparing the application instance identifier received or otherwise associated with the second request to an application instance identifier that Server 106A associated with the resource. The associated application instance identifier may be stored in a local cache or table of server 106A, or it may be stored elsewhere in server cluster 106. If the application instance identifier stored in the cache matches the application instance identifier that is associated with the resource, server 106A invalidates or otherwise frees the resource and allows client 102B to again access the resource on the requestor's behalf without waiting for the resource to be released by some other mechanism (e.g., by the fenced or locked state timing out). If the application instance identifiers do not match, Client 102B will have to wait for the resource to become free before accessing it.

[0026] While in the above example Client 102B reconnected to the same server 106A, it is also possible, in other embodiments, for the client to connect to another node in server cluster 106. For example, client 102B may reconnect to server 106B and submit a second request to regain access to the resource on behalf of the requestor. The second request may again be associated with the requestor's application instance identifier, for example, by being included in the second request or otherwise associated with the second request. In this example, Server 106B may not have the application instance identifier associated with the resource stored in its local cache because the original access of the resource was on server 106A. In such a situation, server 106B may contact the other servers in server cluster 106 to determine if they have an application identifier associated with the resource. If the application identifier associated with the resource is stored on a different node in the server cluster (e.g., server 106A), the application instance identifier on the other node in the server cluster is compared with the application instance identifier provided with the second request. If they match, server 106B may send a request to server 106A to invalidate the resource, and then server 106B may allow the requestor (now on client 102B) to access the resource. If the application instance identifiers do not match, Client 102B will have to wait for the resource to free.

[0027] Based on the above examples, one of skill in the art will appreciate that any client node in the client cluster 102 may request to access for, and then provide access to, a requestor in the client cluster 102. Furthermore, any server node in a server cluster (e.g.,

any server in server cluster 106) is capable of determining whether the requestor previously had access to the resource even if the access occurred on a different server node in the server cluster. One of skill in the art will appreciate that the following description is merely one example of how the embodiment shown in FIG. 1 may operate and other
5 embodiments exist. For example, rather than accessing resources on a remote server or server cluster, a client nodes may perform the embodiments described herein to provide requestors (e.g., applications or processes) continuous access to resources residing in the clustered environment (e.g., on the same or different client cluster nodes that make up the client cluster). As described in greater detail below, embodiments described herein may
10 involve various different steps or operations. Furthermore, the embodiments described herein may be implemented using any appropriate software or hardware component or module.

[0028] Turning now to FIG. 2, the figure illustrates a block diagram of a software environment 200 showing client node cluster 201 with multiple client nodes (e.g., clients
15 202 and 204) and a server node cluster 206 with multiple server nodes (e.g., Node 1 208 and Node 2 216). In embodiments, client 202 requests to access a resource, such as resource 226, in a server cluster environment 206 on behalf of a requestor. Client node cluster 201 may be a client cluster such as client cluster 102 (FIG. 1). Although not illustrated, client cluster may contain more than two clients. Server node cluster 206, may
20 be a server cluster, such as server cluster 106 (FIG. 1) or it may be any other type of clustered environment such as, but not limited to, a virtual network. Resource 226 may be stored in a datastore 228 that is part of the clustered environment. Although not shown, in alternate embodiments, the datastore 228 may not be part of the clustered environment, but may be connected to the clustered environment over a network. Examples of such a
25 network include, but are not limited to, the Internet, a WAN, a LAN, or any other type of network known to the art. In still further embodiments, datastore may be part of a node (e.g., a device) that is a part of cluster 206.

[0029] Server node cluster 206 may include one or more nodes, such as Node 1 208 and Node 2 216. Although only two node clusters are illustrated in FIG. 2, any number of
30 node clusters may be included in clustered environment 206. In embodiments, node clusters 208 and 216 are capable of receiving a request for, performing an operation on, and/or granting access to resource 226. In embodiments, resource 226 may be a file, an object, an application, data, or any other type of resource stored on or accessible by node in node cluster 206 or by a standalone server.

[0030] In embodiments, a client sends an initial request 222 to clustered environment 206. As illustrated in FIG. 2, the initial request 222 may be sent by client 202 and received by Node 1 208. However, in alternate embodiments, the initial request 222 may be sent by client or any other client node in client cluster 201 and received by Node 2 216 or any other node in server cluster 206. An example requests include, but are not limited to, requests to create, open, or otherwise access a file. Request 222 may be transmitted from the client to the Node cluster via a network such as, but not limited to, the Internet, a WAN, a LAN, or any other type of network known to the art. Initial request 222 may include a request to access a resource, such as resource 226. In embodiments, request 222 may also include an application instance identifier that identifies the requestor that client 202 is making the request on behalf of. In embodiments, initial request 222 may consist of one or more messages. For example, request 222 may be a single message containing both the request and an application instance identifier. In another embodiment, request 222 may be multiple messages that include one or more requests as well as one or more application instance identifiers. In embodiments, client 202 may include an App Instance Cache 214 that is used to store and/or generate one or more application instance identifiers that may be transmitted with request 222.

[0031] As shown in FIG. 2, Node 1 208 may receive request 222 and an application instance identifier from client 202. If the requested resource 226 is available, e.g., not fenced or locked by another client or application, Node 1 may grant the client's (e.g., client 202) request to access resource 226 on behalf of a requestor that is executing on the client. Upon granting access to resource 226, filter driver 210 may allocate or otherwise create an association between client 202 and the resource 226 by storing the application instance identifier it received from client 202. In embodiments, the association may be stored in as an object in app instance cache 212 that is a part of Node 1. Although the illustrated embodiment shows app instance cache 212 as a part of Node 1 208, in embodiments, app instance cache 212 may be stored elsewhere as a part of node cluster 206. One of skill in the art will appreciate Node cluster 206 may include one or more app instance caches, such as app instance cache 220 on Node 2 216. In embodiments when more than one app instance cache is present, the data stored in the multiple app instance caches may be replicated across all app instance caches, or each app instance cache may store separate data.

[0032] In one embodiment, the application instance identifier received from client an application instance identifier that identifies a requestor (e.g., an application or process)

may be stored in a `_NETWORK_APP_INSTANCE_ECP_CONTEXT` structure. The `_NETWORK_APP_INSTANCE_ECP_CONTEXT` structure may be defined as follows:

```
typedef struct _NETWORK_APP_INSTANCE_ECP_CONTEXT {
    USHORT Size;
    5    USHORT Reserved;
    GUID AppInstanceID;
    } _NETWORK_APP_INSTANCE_ECP_CONTEXT,
    *PNETWORK_APP_INSTANCE_ECP_CONTEXT;
```

[0033] In such embodiments, the variable size may store information related to the size of the structure and the variable `AppInstanceID` may be a unique application instance identifier for a failover cluster client application, such as a requestor executing on client 202. In embodiments, the `_NETWORK_APP_INSTANCE_ECP_CONTEXT`, or another object or variable containing the requestor's application instance identifier may be stored in the globally unique identifier (GUID) cache 214. In embodiments the

`_NETWORK_APP_INSTANCE_ECP_CONTEXT` structure may be sent from a client to a server in association with a request to access a resource (e.g., a create or open request). In one embodiment, the requestor's application instance identifier may be stored in the GUID cache of the client node that the requestor is executing on in the clustered client environment 201. In another embodiment, although not shown in FIG. 2, the client node cluster 201 may have a central repository that stores application instance identifiers. In such an embodiment, multiple client nodes in the client node cluster 201 may access the centralized repository. In yet another embodiment, application instance identifiers may be stored across multiple GUID caches (e.g., GUID cache 214 and GUID cache 216). In such embodiments, the client node cluster 201 may employ a replication algorithm to ensure that the multiple GUID caches contain the same application instance identifiers.

[0034] As previously described, the application instance identifier may be associated with resource 226 while client 202 accesses resource 226 on behalf of a requestor. A server node 206 may store such an association in one or more app instance caches that are part of server node cluster 206, such as app instance caches 212 and 220. In one embodiment, the application instance identifier may be associated with the resource by adding it to an Extra Create Parameter (ECP) list for the resource 226. The ECP list may be stored in an app instance cache that is part of the server node cluster 206, such as app instance caches 212 and 220. In embodiments, when an ECP is received by a server, the server extracts an application instance identifier from the ECP and adds it to a cache to be

associated with a resource, resource handle, etc. As described with respect to storing application instance identifiers in client cluster 201, the application instance identifiers associated with a node may be stored in an individual app instance cache on node in server node cluster 206, in a central repository in server cluster 206, or replicated across multiple app instance caches on multiple nodes in server node cluster 206.

[0035] In embodiments, resource 226 is fenced or locked while a requestor executing on client 202 has access to resource 222, thereby preventing other client or applications from accessing resource 226 and avoiding any potential conflicts. In embodiments, before the requestor completes its operation on resource 226, client 202 experiences an error that causes it to lose connection with the resource. For example, the client may crash, be taken offline, or lose its network connection to server node 208. In such instances, resource 226 may still be in a fenced or locked state because the requestor did not release a lock on the resource, thereby preventing other clients from accessing resource 226.

[0036] When the error occurs to client 202, the requestor may utilize a client failover mechanism 232 to migrate to a new client node (e.g., client 204) in the client cluster 201. One of skill in the art will appreciate that any type of failover mechanism may be employed at client failover 232. In embodiments, the failover mechanism 232 may also include the migration of the requestor's application instance identifier which may have been stored in GUID cache 214 on the now failed client 202. Upon completing the migration, the requestor may attempt to regain access to the resource 202. In embodiments, client 216 may send a second request 224 to Node 1 to request access to resource 226 on behalf of the requestor. However without the continuous access embodiments disclosed herein, when Node 1 208 receives a request to access the resource 226 on behalf of client 204 (the sender of second request 224), it may deny the request because resource 226 is still in a fenced or locked state from the previous access that client 202 made on behalf of the resource. Without the embodiments disclosed herein, Node 1 208 would recognize that the second request to access resource 226 was from a different location (e.g., client 204). Node 1 208 would not be able to determine that the request is for the same requestor that holds the lock on resource 226, and would therefore determine that granting the request would result in a conflict. However, if the same requestor is attempting to access resource 224, there is no issue of conflict and forcing the client to wait for the resource to be freed by the system may result in undue delays.

[0037] The application instance identifier may be used to solve this problem. In embodiments, the second request 224 may also include the application instance identifier

from identifying the requestor that migrated to client 204 during the failover shown at 232. In embodiments, the requestor's application instance identifier may be present in the GUID cache 228 of client 204 prior to the migration of the requestor during the client failover 232. For example, a replication mechanism may have been employed to replicate the requestor's application instance identifier across the nodes in client cluster 201. In another embodiment, the requestor 203 may store its application instance identifier. In yet another embodiment, the requestor's 203 application instance identifier may be migrated during client failover 232.

[0038] As described with respect to request 222, the application instance identifier may be transmitted in the same message as the second request 224 or the second request 224 may be composed of a number of different messages. When the second request is received at the node cluster 206, or an individual node in the cluster, such as Node 1 208, and the receiving server determines that the resource is fenced or locked, a determination is made as to whether the application instance identifier in the second request 224 is the same as the application instance identifier associated with resource 226. In embodiments, Node 2 216 will compare the application instance identifier received with the second request 222 with the application instance identifier that is associated with resource 226. The application identifier associated with resource 220 may be stored in the app instance cache 212 of Node 1 212. In embodiments where multiple app instance caches exist in node cluster 206, the determination may check more than one application instance caches in the node cluster 206. In such embodiments, if a matching application instance identifier is not located in app instance cache 212, Node 1 216 may send a request to Node 2 212 to determine if a matching application instance identifier is located in app instance cache 220.

[0039] In one embodiment, if the application instance identifier received in second request 224 does not match the application instance identifier associated with resource 226 (which may be stored in application instance cache 212 and/or 220), the second request 224 may not be granted until resource 226 is free. However, if a match is found, the receiving server (e.g., Node 1 208) and/or the server node cluster 206 perform actions to grant access to resource 226 without causing undue delay to client 204 and requestor 203. In such instances, node cluster 206 may invalidate the resource 226, thereby removing resource 226 from a fenced or locked state. In embodiments, invalidating a previous access may comprise any action that brings a resource out of a fenced or locked state. One non-limiting example is closing an opened file (e.g., if resource 226 is a file). Once the

previous access is invalidated, the second request 224 to access resource 226 may be granted, thereby providing continuous access to the requestor 203.

[0040] In one embodiment, the node receiving the second request 224, such as Node 1 208 in FIG. 2, may perform the required actions to invalidate the previous access of resource 226 if a different node (e.g., Node 2 216) has access and/or permission to invalidate the previous access. However, in some instances, the node receiving the request may not have access or permission to invalidate the previous access. For example, such an instance may occur if the original request 222 was made to Node 2 216, in which case Node 2 216 may have control over the resource. In such instances, the node receiving the second request 224 may send a request to the controlling node to invalidate the previous access. Once the controlling node has invalidated the previous access, the node receiving the second request 224 may grant the second request 224. In other embodiments, the node receiving the second request 224 may send a request to a different node to grant client 204 and/or requestor 203 (now residing on client 204) access to resource 226.

[0041] The described process avoids undue delay in granting a second request 224 to access a resource 226 from a resource 203 that previously accessed and still holds a lock on resource 226 through the use of application instance identifiers. Furthermore, the application instance identifiers provide the benefit of ensuring that any request granted does not create a conflict on resource 226. For example, if the request was received from a different application, the request will include an application instance identifier that is different from the application instance identifier associated with the resource which would result in the request being denied. Because application instance identifiers are globally unique identifiers, the application instance identifier for different applications will not be the same.

[0042] FIG. 3 is an embodiment of a method 300 that a requestor may employ to gain continuous access to a resource in a client clustered environment. For example, a requestor may be a client, such as client 202 (FIG. 2), that employs the method 300 to access a resource (e.g., resource 226). In embodiments, the resource may reside on a remote machine, such as a server. The server may be a standalone server or part of a clustered environment, such as sever cluster 206 (FIG. 2). Flow begins at operation 302 where a request for a resource is sent to a server. In embodiments, the request may be to access a resource. In embodiments, accessing a resource may comprise opening a file, creating a file, or otherwise accessing or performing an operation on a resource that may be remote to a client. In embodiments, a requestor may operate in a client clustered

environment. In such embodiments, the request sent at operation 302 may be sent from a first client in the client clustered environment.

[0043] Flow continues to operation 304 where an application instance identifier is sent, for example, to a server (e.g., a standalone server or a node in a clustered environment).

5 In one embodiment, the first client that sent the request may also send the application instance identifier on behalf of the requestor. As earlier described, an application instance identifier is a GUID identifying the requestor (e.g., an application, client, or a child process of an application requesting access to a resource). In one embodiment, the application instance identifier may be sent in a message transmitted via a network. The
10 application instance identifier may be transmitted in the same message containing the request in operation 302 or it may be transmitted in a different message. In such embodiments, an object containing the application instance identifier, such as but not limited to the `_NETWORK_APP_INSTANCE_ECP_CONTEXT` described with respect to FIG. 2, may be sent at operation 302.

15 **[0044]** In one embodiment, an interface may be used to send the application instance identifier at operation 304. The interface may be a kernel level interface located on a client or available to a client operating in a client clustered environment. In embodiments, the kernel level interface may be used by the requestor and/or client to send an application instance identifier to a server. The following is a non-limiting example of a kernel level
20 interface that may be employed at operation 304 to send an application instance identifier:

```
#if (NTDDI_VERSION >= NTDDI_WIN8)
//
// ECP context for an application to provide its instance ID.
//
25 typedef struct _NETWORK_APP_INSTANCE_ECP_CONTEXT {
//
// This must be set to the size of this structure.
//
30 USHORT Size;

//
```

```

// This must be set to zero.
//

USHORT Reserved;

//

// The caller places a GUID that should always be unique for a single instance of
// the application.
//

GUID AppInstanceId;

} NETWORK_APP_INSTANCE_ECP_CONTEXT,
*PNETWORK_APP_INSTANCE_ECP_CONTEXT;

//

// The GUID used for the APP_INSTANCE_ECP_CONTEXT structure.
//
// {6AA6BC45-A7EF-4af7-9008-FA462E144D74}
//

DEFINE_GUID(GUID_ECP_NETWORK_APP_INSTANCE, 0x6aa6bc45, 0xa7ef,
0x4af7, 0x90, 0x8, 0xfa, 0x46, 0x2e, 0x14, 0x4d, 0x74);

#endif // NTDDI_Version >= NTDDI_WIN8

```

[0045] Although a specific kernel level interface is provided, one of skill in the art will appreciate that other kernel level interfaces may be employed at operation 304 to send the application instance identifier.

[0046] In another embodiment, an application user interface (API) may be employed at operation 304 to send an application instance identifier. In such embodiment, the requestor and/or client may send an application instance identifier by making a call to the API. The API may be hosted on the client performing the operation 304 (e.g., the first client in a server cluster) or the API may be hosted on another device and accessed by the

requestor or another application or process. The following is a non-limiting example of an API that may be employed at operation 304 to send an application instance identifier:

```
NTSTATUS RegisterAppInstance (
    _in PGUID AppInstance
);
```

[0047] Although a specific API is provided, one of skill in the art will appreciate that other API's may be employed at operation 304. Furthermore, although operation 304 is illustrated as a discrete operation, one of skill in the art will appreciate that sending the application instance identifier may be performed simultaneously with sending the request at operation 302.

[0048] When the requested resource is not locked, the request sent at operation 302 is granted and flow continues to operation 306 where the resource is accessed. As previously described, the server or device controlling the resource may place the resource in a fenced or locked state while the requestor accesses the resource at operation 306. At some point while accessing the resource, an error occurs, such as the errors described with reference to FIG. 2 which causes the client to fail or otherwise lose connection to the resource. The error may cause the client (e.g., the first client in the server cluster) to lose access to the resource before the requestor completes its use of the resource. Under such circumstances, the resource may not be released from its fenced or locked state.

[0049] Flow continues to operation 308, where a failover operation is performed. In embodiments, the failover operation may comprise cloning the requestor and its state to a different client in the client node cluster (e.g., a second client). In embodiments, the requestor's state may be cloned on the second and the requestor may be executed on the second client in a manner such that it can resume execution from the point where the first client failed. In another embodiment, the requestor may be in communication with the first client (rather than executing on) at the time of the first client's failover. In such embodiments, the failover operation may comprise the requestor establishing communications with a second client in the client cluster.

[0050] In embodiments, state information, including but not limited to the requestor's application instance identifier, may be transferred from the first client to the second client. In one embodiment, the first client may send a message including the requestor's application instance identifier and/or the requestor's state information. The application instance identifier and/or state may be sent during the failover process or, in embodiments, may be sent before the first client fails, such as, during a replication process that clone's

information across the clients in a client clustered environment. In another embodiment, the requestor's application instance identifier and/or state information may be stored in a central location or repository in the client clustered network. In such embodiments, the failover process may provide the second client with the location of the requestor's application instance identifier and/or state information. In yet another embodiment, the requestor may maintain its application instance identifier. In such embodiments, the client failover operation may comprise relocating to or otherwise establishing a connection between the requestor and a second client.

[0051] In embodiments, after the client failover operation flow continues to operation

310. At operation 310 a second request for the same resource is sent to the clustered environment. In embodiments, the second request is sent by the second client in the client cluster on behalf of the requestor. The second request may be sent using the same manner as described with respect to the first resource at operation 302. In order to maintain continuous access to the resource and avoid undue delay, flow continues to operation 312 where the application instance identifier is again sent to the clustered environment. The application instance identifier may be sent at operation 308 according to one of the embodiments described with respect to operation 304. In embodiments, because the a different client (e.g., the second client) is sending the second request, the server receiving the request may not be able to identifier the second request as belonging to the same requestor that holds a lock on the resource (e.g., because the request is made from a different machine, a different address etc.) However, by sending the application instance identifiers at operations 304 and 308, the server will be able to identify the requests as belonging to the same requestor, and will grant continuous access to the resource as previously described with respect to FIGs. 1 and 2. Flow continues to operation 314 and the requestor resumes access to the resource. In embodiments, the second client may receive a response to the second request from the server indicating that the server granted the second request. In embodiments, upon receiving the indication, the second client may access the resource on behalf of the requestor.

[0052] FIG. 4 is an embodiment of a method 400 performed by a node in a server

clustered environment to provide continuous access to a resource. Embodiments of the method 400 may be performed by a node such as Node 1 208 (FIG. 2) in a clustered environment, such as node cluster 206 (FIG. 2). In embodiments, method 400 may be performed by a node that has access to a resource. Flow begins at operation 402 where the node receives a request for a resource. In embodiments, a resource may be a file, an

object, a method, data, or any other type of resource that is under the control of and/or may be accessed by the node performing operation 400. An application instance identifier may be received with the request at operation 402.

[0053] Flow continues to decision operation 404 where a determination is made as to whether the resource is in a fenced or locked state. One of skill in the art will appreciate that any manner of determining whether a resource is fenced or locked may be employed at operation 404. If the resource is not in a fenced or locked state, flow branches NO to operation 412 where the request for the resource is granted. In embodiments, granting the request may comprise allowing the requestor access to the resource, performing an operation on the resource on behalf of the requestor, or permitting any kind of access or modification to the resource. For example, granting the request in operation 412 may include opening a file or creating a file.

[0054] If the resource is in a fenced or locked state, flow branches YES from operation 404 to decision operation 406. At decision operation 406, the application instance identifier received with the request at operation 402 is compared to an application instance identifier that is associated with the resource. For example, as describe with respect to FIG. 2, a node may associate an application instance identifier with a resource when a client or application accesses a resource. As described earlier, the association of the application instance identifier of a requestor accessing a resource may be stored on a node, for example, in an app instance cache, as described in various embodiments discussed in FIG. 2. In embodiments, the application instance identifier that is provided in an ECP sent with a request for a resource, for example, in a `_NETWORK_APP_INSTANCE_ECP_CONTEXT` structure, may be added to an ECP list associated with the resource.

[0055] In one embodiment, the association of the application instance resource may reside locally on the node performing method 400. In such instances, the comparison may be made at a local app instance cache resident on the server. However, as discussed with respect to FIG. 2, a clustered environment may contain a number of app instance caches distributed across different nodes. Furthermore, the different application instance caches may each store separate and/or different data. The application identifier associated with the fenced or locked resource may be stored on a different node in the clustered environment. In such instances, operation 406 may include sending a request to a different node to perform the comparison at operation 406. The request may include the application instance identifier received at operation 402.

[0056] If the received application instance identifier is not the same as the application instance identifier associated with the resource, flow branches NO to operation 410. At operation 410 the request to access the resource received at operation 402 is denied. In embodiments, the request may be denied in order to avoid a resource conflict. Because the received application identifier is not the same as the associated application instance identifier, the request to access the resource received at operation 402 is from a different requestor or application. Granting a request to the different client or application, as may be in this case, may cause a conflict situation that will interfere with the application currently accessing the resource. For example, the different application may modify the resource in a manner that modifies or otherwise interferes with the operations performed on the resource by the requestor that currently holds a lock on the resource.

[0057] However, receiving an application identifier with the request 402 that is the same as the application identifier associated with the fenced or locked resources indicates that an error may have occurred that caused the requestor that was accessing the resource to lose its access to the resource without properly releasing the resource. For example, the requestor may operate in a client node cluster. The particular client the requestor was operating on may have lost connection to the server or otherwise failed before the requestor completed its operations upon the resource. In order to provide continuous access the resource, that is, to allow the requestor to regain access to the resource without experiencing undue or unacceptable delay, flow branches YES to operation 408.

[0058] At operation 408, the resource is invalidated. As earlier described herein, invalidating the resource may include changing the fenced state of the resource or otherwise removing a lock on the resource. For example, if the resource is a file, invalidating the resource may include closing the file. One of skill in the art will appreciate that any method of releasing a fenced or locked resource may be employed at operation 408.

[0059] Referring back to FIG. 2, in embodiments, access to a resource may be under control of a node in the clustered environment different than the node that receives the request for access to the resource at operation 402. For example, a handle to the resource may reside on a different node in the clustered environment. In such embodiments, invalidating the resource may include sending a request to the node controlling access to the resource to invalidate the resource. In response to sending the request, the remote node may invalidate the resource.

[0060] After the resource is invalidated, flow continues to operation 412 where the request to access the resource is granted. Granting the request may comprise allowing the requestor access to the resource, performing an operation on the resource on behalf of the requestor, or permitting any kind of access or modification to the resource. For example, granting the request in operation 412 may include opening a file or creating a file. Granting such access may be performed by the node receiving the request at operation 402, or by another node in the clustered environment.

[0061] Methods 300 and 400 are merely some examples of operational flows that may be performed in accordance with embodiments. Embodiments are not limited to the specific description provided above with respect to FIGS. 3-6 and may include additional operations. Further, operational steps depicted may be combined into other steps and/or rearranged. Further, fewer or additional steps may be used, employed with the methods described in FIGs. 3-4.

[0062] FIG. 5 illustrates a general computer system 500, which can be used to implement the embodiments described herein. The computer system 500 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computer system 500 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the example computer system 500. In embodiments, system 500 may be used as the clients and/or servers described above with respect to FIGs. 1 and 2.

[0063] In its most basic configuration, system 500 typically includes at least one processing unit 502 and memory 504. Depending on the exact configuration and type of computing device, memory 504 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination. This most basic configuration is illustrated in FIG. 5 by dashed line 506. System memory 504 stores instructions 520 such as the instructions to perform the continuous availability methods disclosed herein and data 522 such as application instance identifiers that may be stored in a file storage system with storage such as storage 508.

[0064] The term computer readable media as used herein may include computer storage media. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory 504, removable storage, and non-removable storage 508 are all computer

storage media examples (e.g. memory storage). Computer storage media may include, but is not limited to, RAM, ROM, electrically erasable read-only memory (EEPROM), flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store information and which can be accessed by computing device 500. Any such computer storage media may be part of device 500. Computing device 500 may also have input device(s) 514 such as a keyboard, a mouse, a pen, a sound input device, a touch input device, etc. Output device(s) 516 such as a display, speakers, a printer, etc. may also be included. The aforementioned devices are examples and others may be used.

[0065] The term computer readable media as used herein may also include communication media. Communication media may be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term “modulated data signal” may describe a signal that has one or more characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), infrared, and other wireless media.

[0066] Embodiments of the invention may be practiced via a system-on-a-chip (SOC) where each or many of the components illustrated in Figure 5 may be integrated onto a single integrated circuit. Such an SOC device may include one or more processing units, graphics units, communications units, system virtualization units and various application functionality all of which are integrated (or “burned”) onto the chip substrate as a single integrated circuit. When operating via an SOC, the functionality, described herein, with respect to providing continuous access to a resource may operate via application-specific logic integrated with other components of the computing device/system 500 on the single integrated circuit (chip).

[0067] Reference has been made throughout this specification to “one embodiment” or “an embodiment,” meaning that a particular described feature, structure, or characteristic is included in at least one embodiment. Thus, usage of such phrases may refer to more than just one embodiment. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0068] One skilled in the relevant art may recognize, however, that the embodiments may be practiced without one or more of the specific details, or with other methods, resources, materials, etc. In other instances, well known structures, resources, or operations have not been shown or described in detail merely to avoid obscuring aspects of the embodiments.

[0069] While example embodiments and applications have been illustrated and described, it is to be understood that the embodiments are not limited to the precise configuration and resources described above. Various modifications, changes, and variations apparent to those skilled in the art may be made in the arrangement, operation, and details of the methods and systems disclosed herein without departing from the scope of the claimed embodiments.

Claims

1. A method of providing continuous access to a resource, the method comprising:
receiving a first request to access a resource from a requestor, wherein the request
is received from a first client;
5 associating a first application instance identifier with the first resource;
allowing the first request access to the resource;
receiving a second request for the resource from the requestor after recovering
from a failure, wherein the second request is received from a second client different from
the first client;
10 receiving a second application identifier associated with the second request;
determining if the first application identifier and the second application identifier
are the same; and
when the first and second application identifiers are the same, performing the steps
comprising:
15 invalidating the first request; and
granting the second request to access the resource.
2. The method of claim 1, wherein the first application identifier is associated with an
application instance of an open request.
3. The method of claim 1, wherein the first application identifier is associated with a
20 process.
4. The method of claim 1, wherein the first application identifier is associated with at
least one child process of an application.
5. The method of claim 1, wherein associating the first application instance identifier
comprises receiving the first application instance identifier in a
25 NETWORK_APP_INSTANCE_ECP_CONTEXT structure.
6. A method for providing clustered client failover, the method comprising:
receiving, at a second client, an application instance identifier for a requestor,
wherein the requestor previously accessed a resource using a first client;
sending, from the second client, a second request to access the resource on behalf
30 of the requestor;
sending, from the second client, the application instance identifier for the
requestor;
receiving an indication that a server granted the second request; and
accessing, by the second client, the resource on behalf of the client.

7. The method of claim 6, wherein the server granting the second request previously granted a first request to access the resource from the first client on behalf of the requestor.

8. The method of claim 6, wherein the second client sends the second request in response to a client failover.

5 9. A system for facilitating client failover in a clustered environment, the system comprising:

at least one server comprising:

at least one processor configured to execute computer executable instructions;

10 at least one computer readable storage media storing the computer executable instructions that when executed by the at least one processor provide:

receiving a first request to access a resource from a first client on behalf of a requestor;

15 associating a first application instance identifier with the first resource;

allowing the requestor access to the resource;

receiving a second request for the resource from a second client, wherein the second client is different from the first client;

20 receiving a second application identifier associated with the second request;

determining if the first application identifier and the second application identifier are the same;

when the first and second application identifiers are the same, performing the steps comprising:

25 invalidating the first request; and

granting the second request to access the resource.

10. The system of claim 18, wherein the system further comprises: the first client, comprising:

30 at least one processor configured to execute computer executable instructions;

at least one computer readable storage media storing the computer executable instructions that when executed by the at least one processor:

sending the first request;

sending the application instance identifier to a second client.

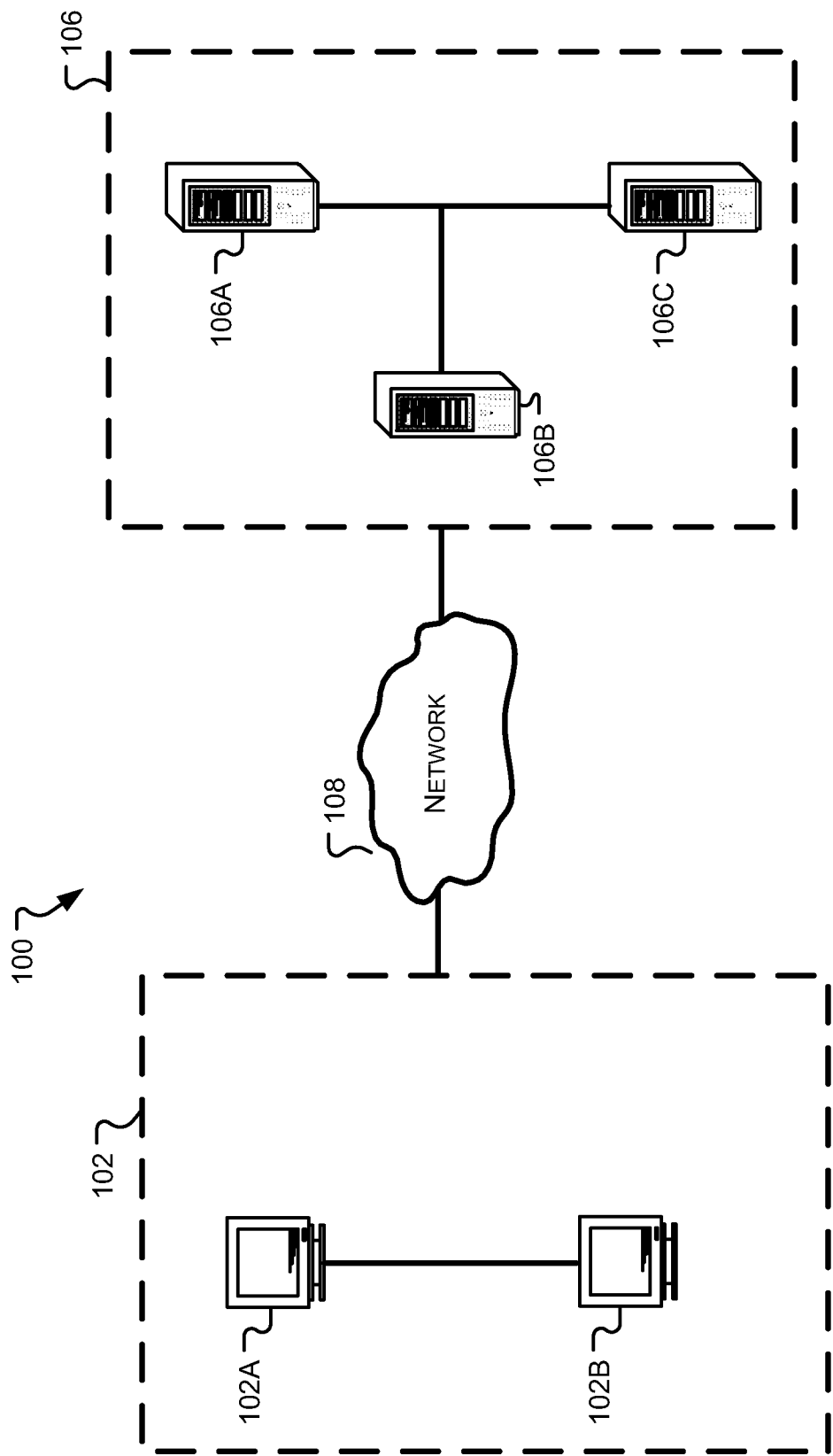


FIG. 1

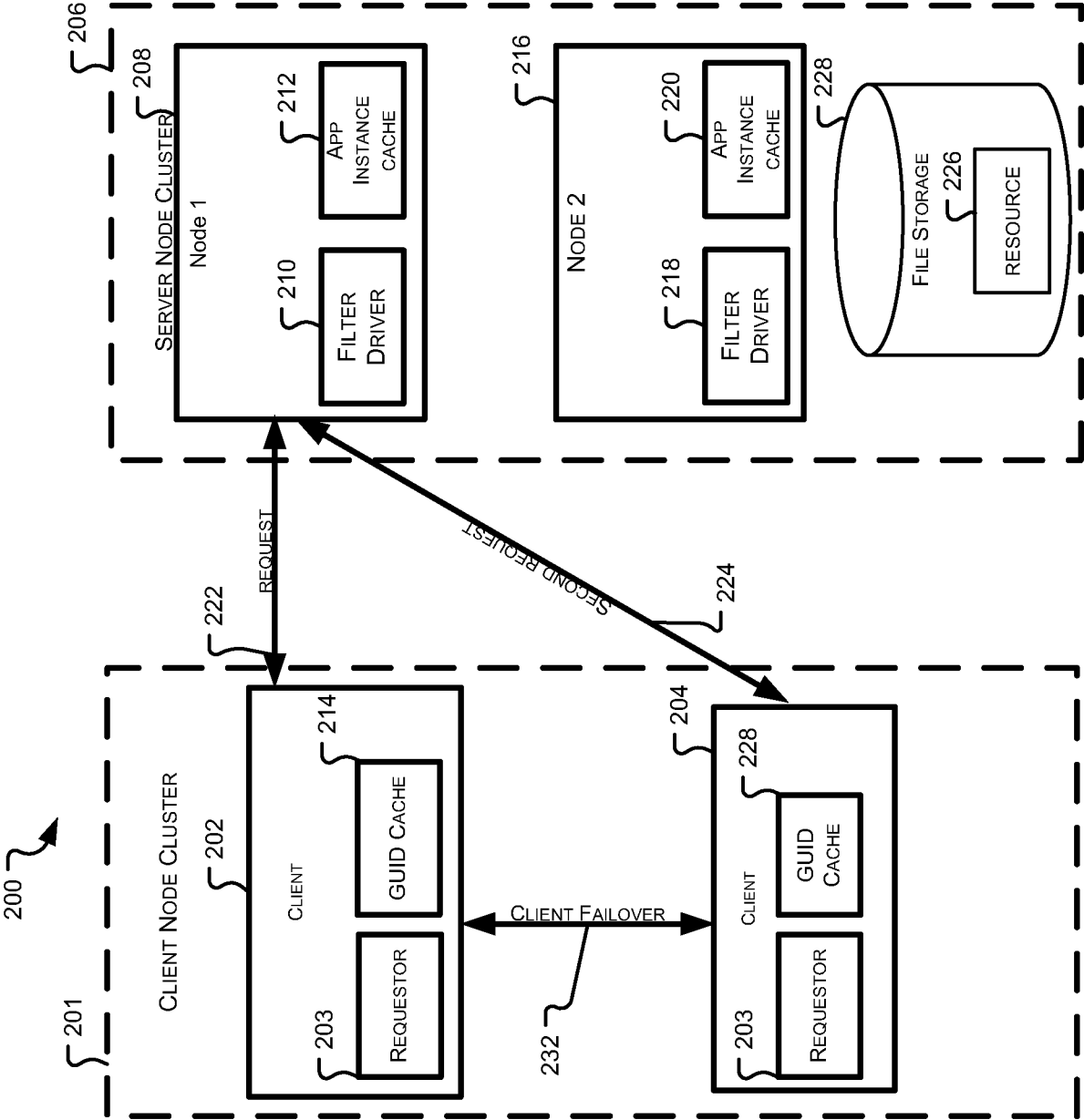
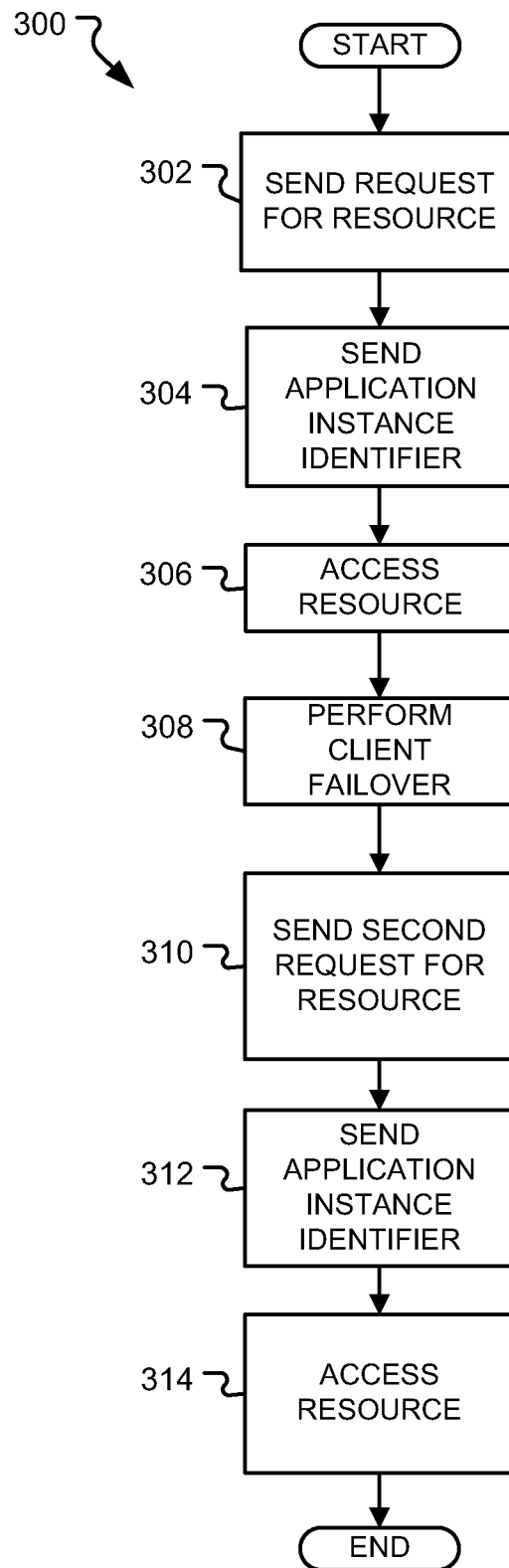
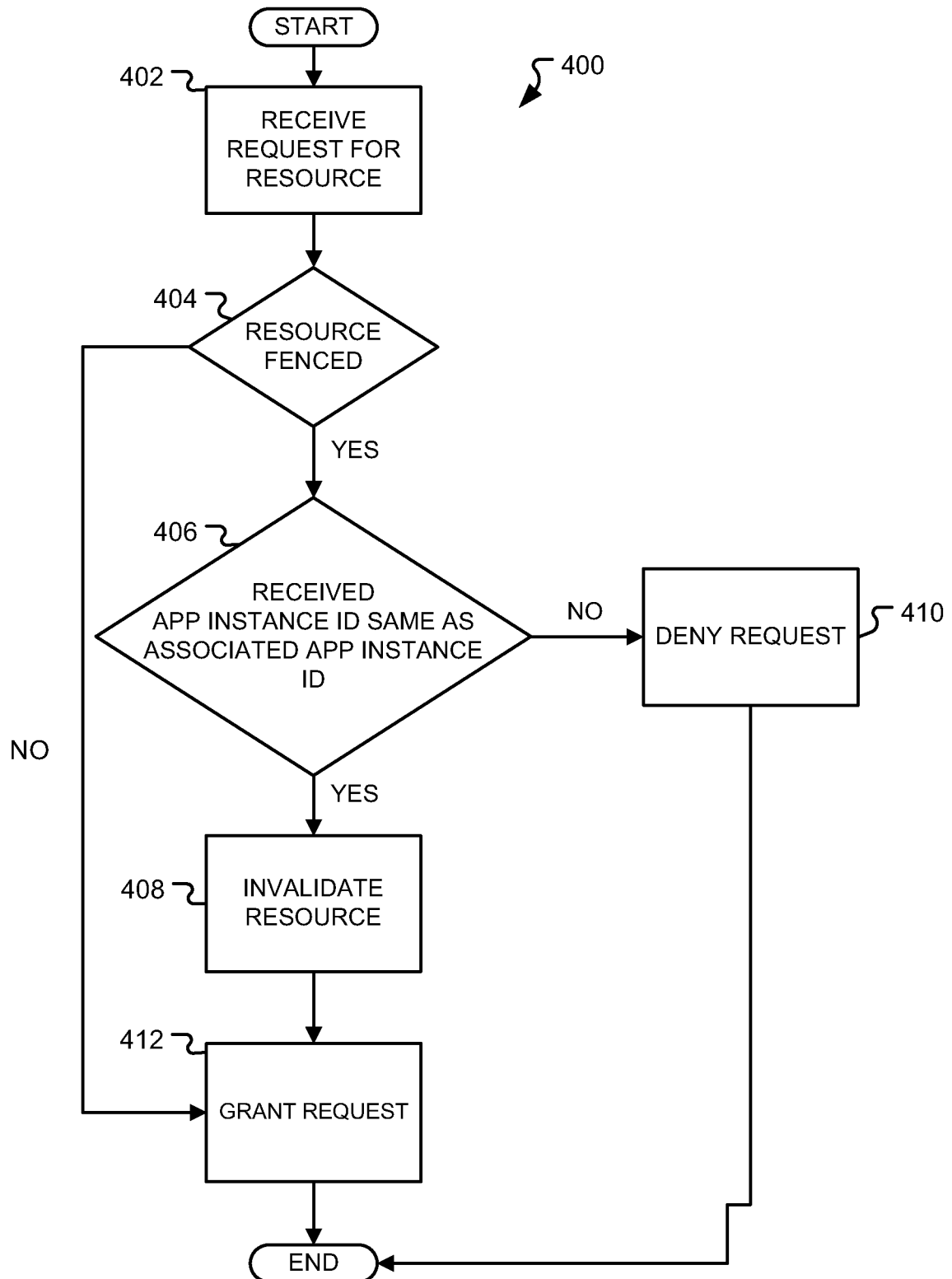


FIG. 2

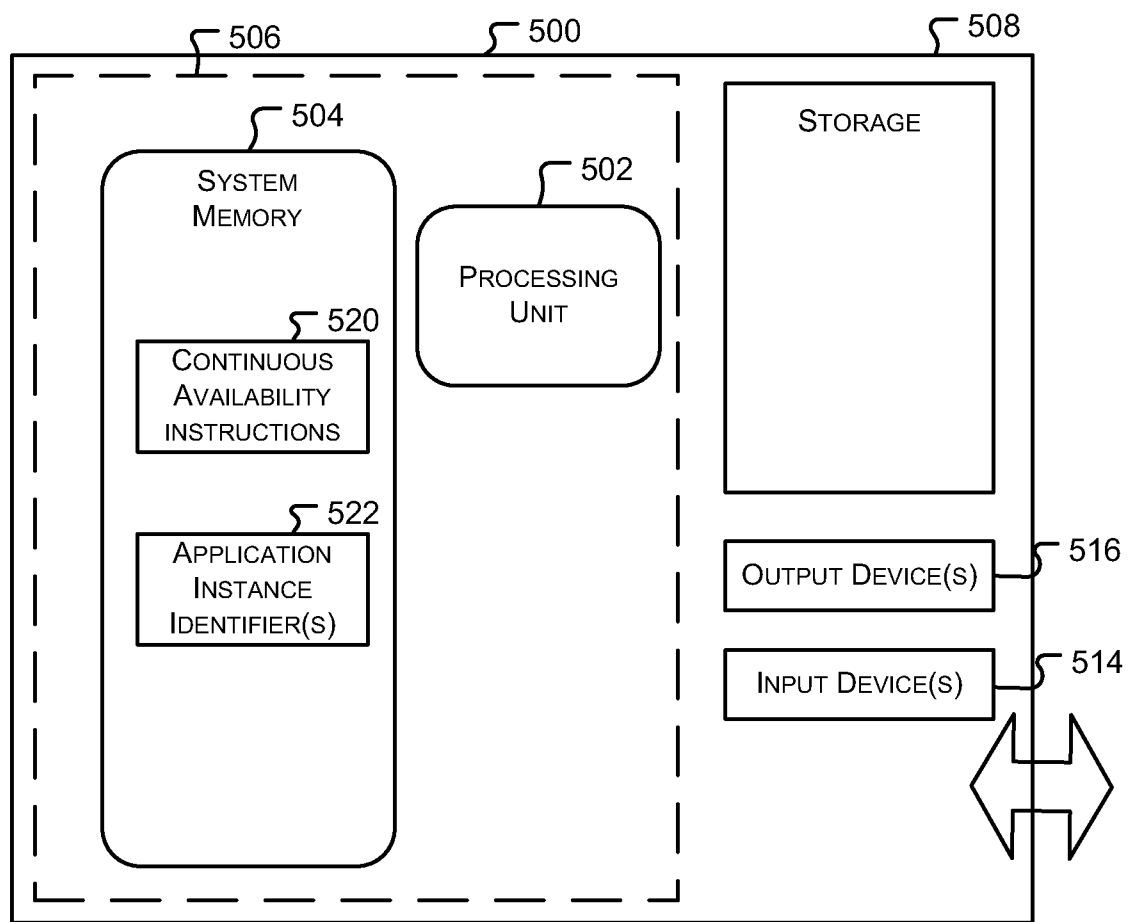
3/5

**FIG. 3**

4/5

**FIG. 4**

5/5

**FIG. 5**



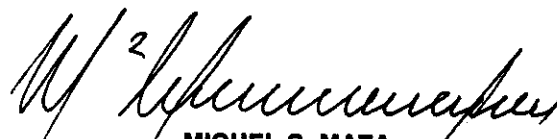
000333

CONSULADO GENERAL DE COLOMBIA
595 Market Street
Suite 2130

San Francisco, California MAYO. 30 / 1995 No. 969 4/23

El suscrito Consul General de Colombia CERTIFICA que el Señor
RALPH MUNRO, quien autoriza el
presente documento, ejercia legalmente, en la fecha alli expresada,
las funciones de SECRETARIO DE ESTADO -
ESTADO DE WASHINGTON, y que la firma y sello que
en el documento aparecen como suyos son los que usa y acostumbra
en sus actos oficiales. El Consulado no asume responsabilidad
alguna por el contenido del documento.




MIGUEL S. MAZA
CONSUL GENERAL DE COLOMBIA

PAGADO IMPUESTO DE TIMBRE
DERECHOS CONSULARES

JEFE DE LEGALIZACIONES
SANTAFE DE BOGOTA D.C.
1995 JUN 12 A 11: 06
RESPONSABILIDAD DEL TEXTO
AD SE ADOPTA

MINISTERIO DE RELACIONES
EXTERIORES
1995 JUN 12 A 11: 06
RESPONSABILIDAD DEL TEXTO
AD SE ADOPTA
SANTAFE DE BOGOTA D.C.
JEFE DE LEGALIZACIONES

STATE of WASHINGTON



SECRETARY of STATE

I, **RALPH MUNRO**, Secretary of State of the State of Washington and custodian of its seal, hereby certify that, according to the records on file in my office, **CAROLYN M. SCHUSSLER** is a Notary Public in and for the state of Washington.



DATE: May 22, 1995



Given under my hand and the seal of the State of Washington at Olympia, the State Capital

RALPH MUNRO

Ralph Munro, Secretary of State

Ind. off. # 4697/95
INES DE HODSON
 Intérprete Oficial
 Resolución No. 2190
 de Minjusticia

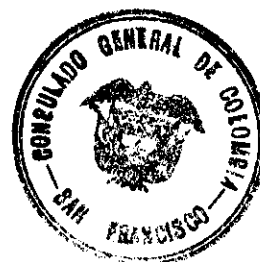
POWER OF ATTORNEY

In the city of Redmond, Washington
on the 12 day of the month of May, one thousand nine
hundred and ninety five (1995), before
me, Carolyn M. Schusser Notary Public,
appeared David Curtis of age, and domiciled
at Seattle, Washington, and
stated:

FIRST: That he/she acts in his/her capacity of Assistant Secretary of
MICROSOFT CORPORATION a company legally constituted and in
existence in accordance with the laws of the State of Washington, United
States of America, and domiciled at One Microsoft Way, Redmond,
Washington 98052-6399, United States of America as evidenced in the
By Laws, documents which I have examined.

SECOND: That in such capacity he/she grants hereby special, ample
and sufficient Power of Attorney to DARIO CARDENAS-NAVAS and/or
EDUARDO CARDENAS-CABALLERO and/or BERNARDO P.
CARDENAS-MARTINEZ y/o JAMES VALDIRI-REYES and/or LUZ
HELENA ADARVE-GOMEZ, registered lawyers of the firm CARDENAS
& CARDENAS-LAWYERS, domiciled at Carrera 7 No. 71-52, Torre B,
Piso 9 Bogotá, D.C., Colombia, to carry before the corresponding
Colombian national Offices and authorities the proceedings to obtain
trademarks, patents, utility models, industrial drawings and designs,
deposits of commercial names and ensigns, as well as processing of
renewals, assignments, change of name or domicile of the owner,
waivers and licenses of the same for which purpose they are empowered
to take before said authorities all necessary actions for the indicated
purposes; to file applications, statements, claims, formulate descriptions,
amendments, oppositions and appeals; pay all taxes and installments
and make any other payments determined by law; to receive every
document and values issuing the respective discharge; to fulfill any other
requirements and take, in general, every measure that they shall deem
opportune for the defense and protection of the Industrial Property of the
company in the event of infringements, oppositions, proceedings for
cancellation or annulment, imposition of fines, passing the antecedents

J. D. 4692/95
INES DE HODSON
Instituto Oficial
Registración No. 2196
de Magistratura



to the courts; they are empowered to appear as plaintiff or defendant before the competent judges, being entitled to transact, submit to arbitration, waive, receive, resign, appeal and all other legal powers which may be necessary, and it is hereby declared ever since now valid and good whatever such attorneys do for its benefit, granting them also power to substitute this proxy and if necessary to revoke such substitutions.

THIRD: That for purposes of articles 597 and 543 of the Colombian code of Commerce the attorneys are empowered to receive service of notice and designate judicial and extrajudicial attorneys.

SIGNED AT Redmond, Washington, United States of America

ON May 13 1995

BY

David Curtis, Assistant Secretary

Carolyn M. Schussler
NOTARY PUBLIC

(To be legalized up to the Colombian Consul

J.O. 4097/95-
INES DE HODSON
Inferencia Oficial
Resolución No. 2198
de Manizales



The undersigned Notary Public certifies that Mr. David Curtis
(nationality), of legal age, domiciled at Seattle, Washington
(civil status) appeared before me and signed the preceding
document; that I know him personally and I am certain that he has
legal capacity to grant it; that he is Assistant Secretary
of MICROSOFT CORPORATION and is consequently
authorized to grant such document; that the named company exists
and is organized under the laws of the state of Washington in conformity with the laws
of the U.S.A. and is domiciled in Redmond, Washington and that
the act for which this document is granted is included among
those which constitute the corporate object of the named company.

I am certain of all these facts because I have examined the
respective documents consisting of the Bylaws, all
of which I hereby attest.

Issued and signed in Redmond, Washington

Dated May 17, 1995

Carolyn M. Schussler

Legalization by Colombian Consul is required. The Colombian
Consul must certify that the company exists and practices its
corporate purpose in accordance with the laws of the respective
country and that the grantor is its representative

ECC/las

F.O. 4692/95
INES DE HODSON
Intendente Oficial
Resolución No. 2199
de 1994



TRADUCCION OFICIAL N°4697/95 de Poder escrito en idioma inglés, el cual para su identificación se sellan con el sello del traductor a tiempo con la presente.

=====

P O D E R

En la ciudad de Redmond, Washington, el día 12 de mayo de mil novecientos noventa y cinco (1995), ante mí, Carolyn M. Schussler, Notario Público, compareció David Curtis, mayor de edad y comociliado en Seattle, Washington, y manifestó:

PRIMERO: Que actúa en su calidad de Subsecretario de MICRO-SOFT CORPORATION, compañía legalmente constituida y existente de acuerdo con las leyes del Estado de Washington, Estados Unidos de América, y domiciliada en One Microsoft Way, Redmond, Washington 98052-6399, Estados Unidos de América, según consta en los Estatutos, documentos que he examinado.

SEGUNDO: Que en tal calidad, por el presente otorga Poder especial, amplio y suficiente a DARIO CARDENAS NAVAS y/o EDUARDO CARDENAS CABALLERO y/o BERNARDO P. CARDENAS y/o JAMES VALDIRI REYES y/o LUZ HELENA ADARVE GOMEZ, abogados titulados de la firma CARDENAS & CARDENAS - ABOGADOS, domiciliada en la Carrera 7 N° 71-52, Torre B, Piso 9, Bogotá, D.C., Colombia para que adelanten ante las oficinas y autoridades nacionales colombianas correspondientes los trámites para obtener marcas de fábrica, patentes, modelos de utilidad, dibujos y diseños industriales, depósitos de nombres comerciales y emblemas, así como el trámite de renovaciones, cesiones, cambio de nombre o domicilio del propietario, desistimientos y licencias de los mismos, a cuyo propósito están facultados para tomar ante dichas autoridades todos los pasos necesarios para los propósitos indicados; presentar solicitudes, declaraciones, reclamos, formular descripciones, enmiendas, oposiciones y apelaciones; pagar todos los impuestos

y cuotas y hacer cualesquiera otros pagos determinados por ley; recibir todos los documentos y valores expidiendo el descargo respectivo; llenar cualesquiera otros requisitos y tomar, en general, todas las medidas que consideren oportunas para la defensa y protección de la propiedad industrial de la compañía en caso de infracciones, oposiciones, trámites para cancelación o anulación, imposición de multas, pasando los antecedentes a los tribunales. Están facultados para comparecer como demandante o demandado ante los jueces competentes, teniendo derecho a transar, someter a arbitramento, renunciar, recibir, desistir, apelar y todos los demás poderes legales que sean necesarios, y desde ahora se declara válido y bueno cuanto dichos apoderados hagan en beneficio de la compañía, otorgándoles también poder para sustituir este mandato y, si es necesario, revocar tales sustituciones.

TERCERO: Para los fines de los artículos 597 y 543 del Código de Comercio de Colombia, los apoderados están facultados para recibir notificaciones y designar agentes judiciales y extrajudiciales.

Firmado en Redmond, Washington, Estados Unidos de América
el 12 de mayo de 1995

Por (fdo) David Curtis, Subsecretario

(fdo) Carolyn M. Schussler, Notario Público.

(Sello notarial en relieve).

(Debe legalizarse hasta por Cónsul colombiano)

El suscrito Notario Público certifica que el Sr. David Curtis (nacionalidad) mayor de edad, domiciliado en Seattle, Washington (estado civil) compareció ante mí y firmó el documento precedente; que lo conozco personalmente y me consta que tiene capacidad legal para otorgarlo; que es Subsecretario de MICROSOFT CORPORATION Y EN CONSECUENCIA ESTA AUTORIZADO PARA

Y. J. HOPSON
Notario Público
Lic. No. 2180
en la jurisdicción

otorgar dicho documento; que la mencionada compañía existe y está constituida según las leyes del Estado de Washington de conformidad con las leyes de los E.U.A. y domiciliada en Redmond, Washington, y que el acto para el cual se otorga este documento está incluido entre aquellos que constituyen el objeto social de la mencionada compañía.

Estoy seguro de todos estos hechos porque he examinado los respectivos documentos consistentes en los Estatutos, de todo lo cual doy aquí fé.

Expedido y firmado en Redmond, Washington

Fecha: el 12 de mayo de 1995

(fdo) Carolyn M. Schussler (Sello notarial en relieve)

Se requiere legalización por Cónsul colombiano. El Cónsul colombiano debe certificar que la compañía existe y ejerce su objeto social de acuerdo con las leyes del respectivo país y que el otorgante es su representante.

- - - - -

ESTADO DE WASHINGTON

SECRETARIO DE ESTADO

Yo, RALPH MUNRO, Secretario de Estado del Estado de Washington y guardián de su sello, por el presente certifico que se acuerdo con los registros de mi oficina CAROLYN M. SCHUSSLER es un Notario Público en y para el Estado de Washington.

Fecha: Mayo 22 de 1995

Dado bajo mi firma y el sello del Estado de Washington en Olympia, la Capital del Estado.

(fdo) Ralph Munro - Secretario de Estado

(Sello del Estado de Washington)

En español, legalizaciones por el Consulado General de Colombia en San Francisco, California, y hasta por el Ministerio de Relaciones Exteriores de Colombia bajo el N°135293 de junio 12 de 1995.

Es traducción fiel y completa del documento adjunto.

Santa Fe de Bogotá, D.C., junio 13 de 1995

Inés de Hobson
INES DE HOBSON

Intérprete Oficial

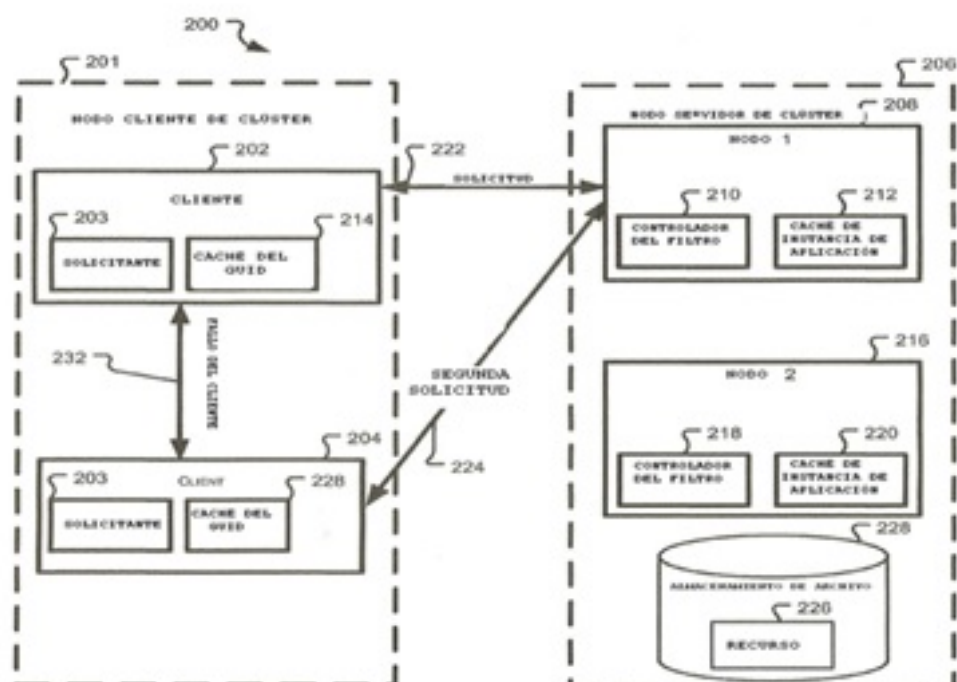
Resolución No. 2190

de Minjusticia

cc. 00054636 000

MINISTERIO DE RELACIONES
EXTERIORES
11/06/95
167131
FOLIO 127
FOLIO 127
DOCUMENTO DESEMPENA
LAS FUNCIONES INDICADAS

NO SE ASUME
RESPONSABILIDAD DEL TEXTO
1995 JUN 18 A 11:28
JEFE DE LEGALIZACIONES
SANTA FE DE BOGOTÁ D.C.



FALLO DEL CLIENTE EN CLÚSTER

RESUMEN

Un identificador de instancia de la aplicación es empleado con varios sistemas y métodos, de modo tal, de proveer un solicitante con acceso continuo a un recurso cuando está en operación en un ambiente de cliente en clúster. Un solicitante que reside en un primer cliente, puede intentar acceder a un recurso. El primer cliente envía una solicitud para acceder al recurso. La solicitud puede estar asociada con un identificador de instancia de la aplicación que identifica al solicitante. En algún momento, el primer cliente falla y el solicitante está asociado con un segundo cliente a través de un mecanismo de fallos. El segundo cliente envía una segunda solicitud para acceder al recurso por cuenta del solicitante. La segunda solicitud está asociada con el identificador de ejemplo de la aplicación del solicitante. El identificador de ejemplo de la aplicación se usa para identificar la segunda solicitud como perteneciente al mismo solicitante que la primera solicitud, otorgando de ese modo el acceso de la segunda solicitud al recurso, en tanto evita una situación de conflicto.

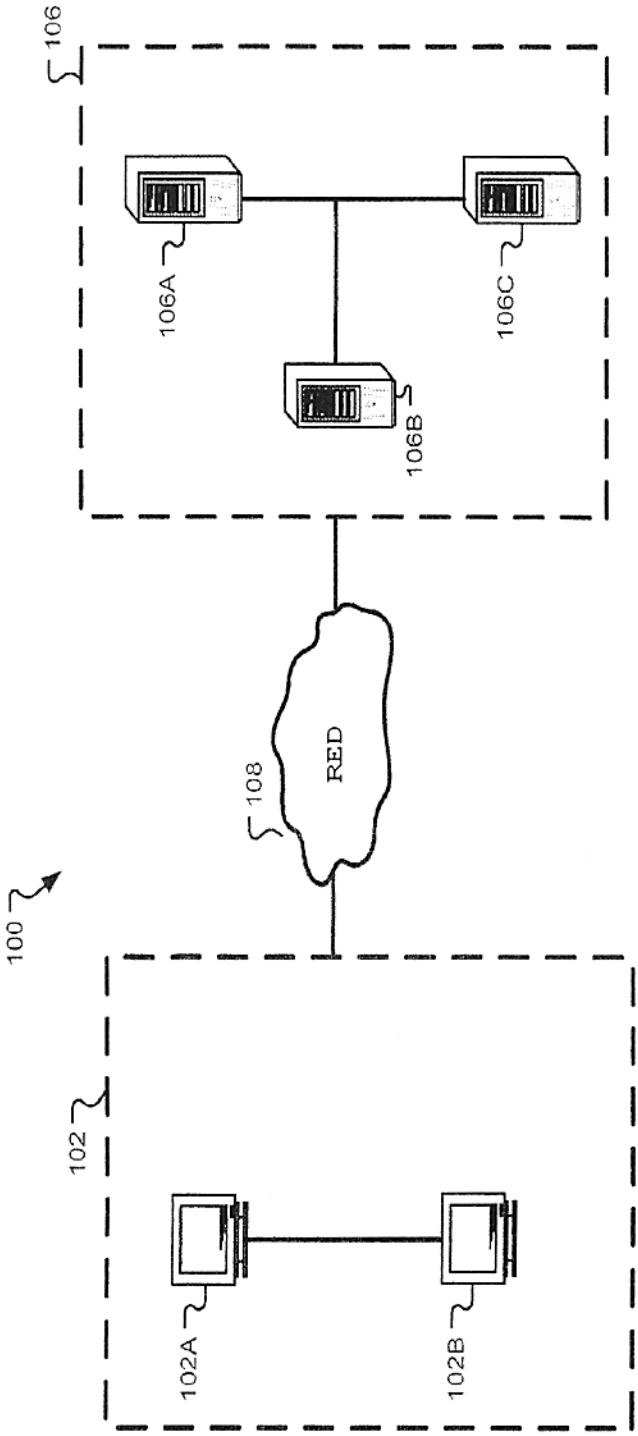
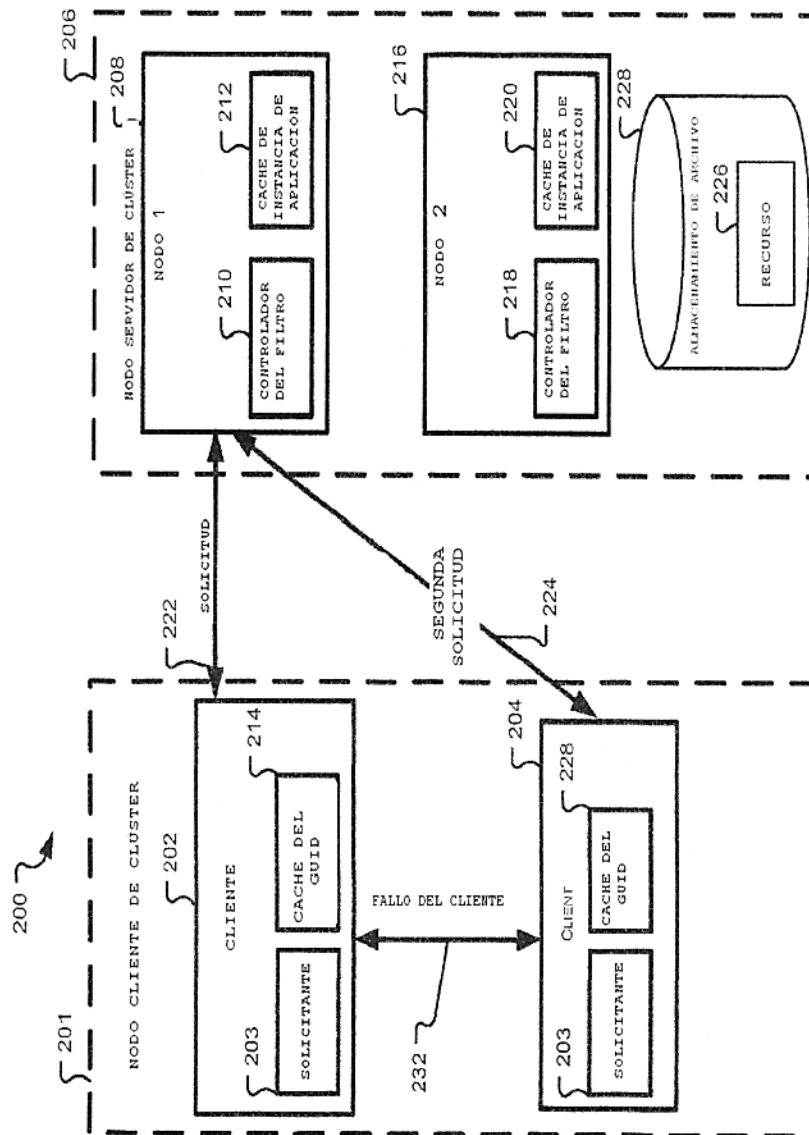


FIG. 1



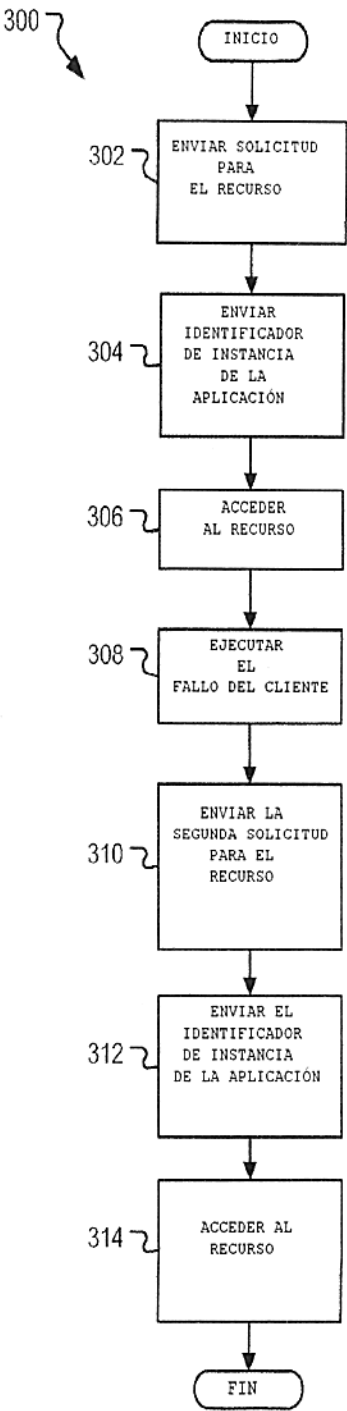


FIG. 3

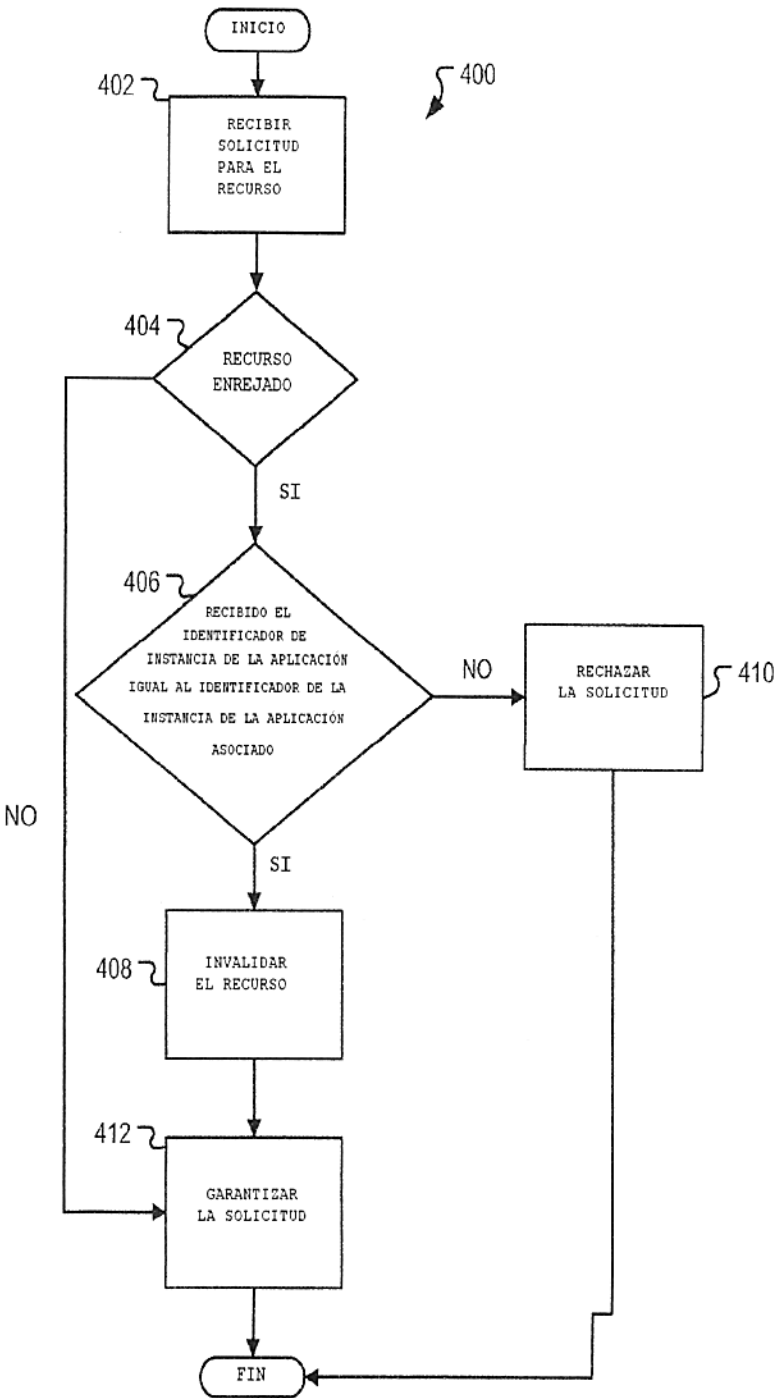


FIG. 4

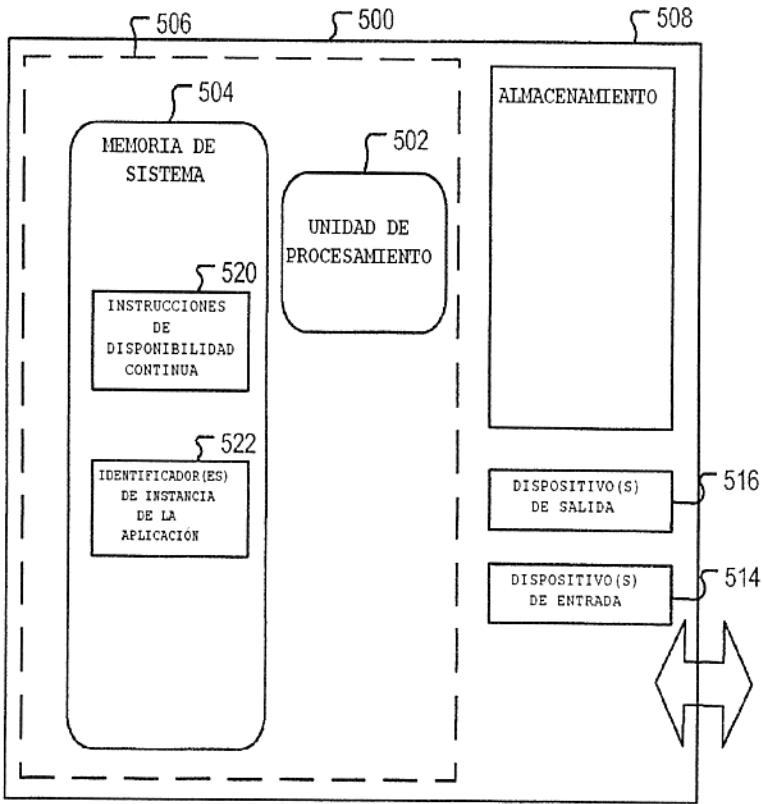


FIG. 5

MEMORIA DESCRIPTIVA

Los ambientes en clúster, por ejemplo, los ambientes en donde la carga de trabajo está distribuida a través de múltiples máquinas, se usan normalmente para proveer fallas y alta disponibilidad de información a los clientes. Los ambientes en clúster les permiten a los clientes acceder a los recursos a través de uno o más nodos que son una parte del ambiente. Un ambiente en clúster puede actuar como un cliente, un servidor, o ambos. En un servidor en clúster de cliente, una aplicación puede residir en cualquiera de los nodos que componen el clúster. La aplicación puede emitir solicitudes para los recursos que son almacenados en forma local dentro del clúster cliente o almacenados en forma remota. Si ocurre un error en el nodo, el cliente falla, o migra a un nodo diferente en el clúster. Sin embargo, cuando el cliente solicita nuevamente el acceso a un recurso con el que estaba trabajando en el momento del error, el recurso puede ser enrejado o cerrado por el servidor para el nodo cliente anterior en que la aplicación residía.

Con respecto a estas y a otras consideraciones es que se han efectuado las realizaciones. Más aún, aunque se han discutido problemas relativamente específicos, se deberá entender que las realizaciones no deberán estar limitadas a resolver los problemas específicos identificados en el ambiente.

Resumen

Se provee este resumen para introducir una selección de conceptos en una forma simplificada, los que están más ampliamente descritos más adelante en la sección Descripción Detallada. Este resumen no pretende identificar características clave o características esenciales de la materia en referencia reivindicada, ni tampoco pretende ser

usado como una ayuda en la determinación del ámbito de la materia en referencia reivindicada.

Los sistemas y métodos se describen aquí para proveer una aplicación o un proceso con acceso continuo a un recurso, después que la aplicación migra a un nuevo nodo en un ambiente de cliente en clúster. Una aplicación o proceso que reside en un nodo en un clúster del cliente, envía una solicitud a un servidor para acceder a un recurso. En las realizaciones, se usa un identificador de instancia de la aplicación único para identificar una aplicación que solicita un recurso. El identificador de aplicación único puede ser provisto con la solicitud. Cuando el cliente accede a un recurso, el identificador de instancia de la aplicación está asociado con el recurso solicitado.

Antes de que la aplicación o el proceso complete sus operaciones en el recurso, el nodo sobre el cual reside el cliente en el ambiente en clúster puede experimentar un error que lo hace fallar o, de otro modo, perder el acceso al recurso antes de que la aplicación libere en forma apropiada el recurso. En tales circunstancias, el recurso puede permanecer en un estado enrejado o cerrado en el servidor por la solicitud del cliente anterior. Al fallar a un nodo diferente en el clúster del cliente, la aplicación en el nuevo nodo cliente puede restablecer una conexión con el servidor que administra el recurso y efectuar una segunda solicitud para el recurso, a la cual tuvo acceso previamente la aplicación en el momento del error. La segunda solicitud puede incluir un identificador de instancia de la aplicación que fue enviado con la primera solicitud. Aunque la segunda solicitud para el recurso puede ser recibida desde un nodo diferente en el ambiente en clúster, el identificador de instancia de la aplicación le permite al servidor que administra la solicitud,

determinar que la segunda solicitud pertenece a la misma aplicación o proceso que cerró anteriormente el recurso. Al hacer esto, se le permite al servidor invalidar el recurso y garantizar que la segunda solicitud del cliente acceda al recurso, en tanto asegura que no surgirá una situación de conflicto.

Las realizaciones pueden ser implementadas como un proceso de computador, como un sistema de computación o como un artículo de elaboración, tal como un producto de programa de computador o medio legible por computador. El producto de programa de computador puede ser un medio de almacenamiento por computador legible por un sistema de computador y que codifica un programa de computador de instrucciones para ejecutar un proceso de computador. El producto de programa de computador además puede ser una señal propagada en un portador legible por un sistema de computación y que codifica un programa de computador de instrucciones para ejecutar un proceso de computador.

Breve descripción de los dibujos

Las realizaciones no limitantes y no exhaustivas se describen con referencia a las siguientes figuras.

La Fig. 1 ilustra un sistema que puede ser usado para implementar las realizaciones descritas aquí.

La Fig. 2 es un diagrama de bloques que ilustra un ambiente de software que puede ser usado para implementar las realizaciones descritas aquí.

La Fig. 3 es una realización de un método que un cliente puede ejecutar, para ganar el acceso continuo a un recurso en un ambiente en clúster.

La Fig. 4 es una realización de un método ejecutado por un nodo en un ambiente en clúster, para proveer el acceso continuo a un recurso.

La Fig. 5 ilustra un diagrama de bloques de un ambiente de computación adecuado para implementar las realizaciones.

Descripción detallada

Se describen más ampliamente varias realizaciones más adelante con referencia a los dibujos que se acompañan, los cuales forman una parte de este documento, y los que muestran realizaciones específicas a modo de ejemplo. Sin embargo, las realizaciones puede ser implementadas en muchas formas diferentes y no deberán ser interpretadas como limitadas con respecto a las realizaciones establecidas aquí, más bien, se proveen estas realizaciones de manera tal de que esta descripción sea detallada y completa, y expresará completamente el ámbito de las realizaciones para aquellas personas entendidas en la materia. Las realizaciones pueden ser practicadas como métodos, sistemas o dispositivos. De acuerdo a lo anterior, las realizaciones pueden tomar la forma de una implementación de hardware, de una completa implementación de software o de una implementación que combine aspectos de software y de hardware. Por lo tanto, la siguiente descripción detallada no deberá tomarse en un sentido limitante.

Las realizaciones de la presente descripción están relacionadas para proveer mecanismos de fallo del cliente en clúster que le permitan a un solicitante recuperar el acceso a un recurso después de un evento de fallo. En las realizaciones, un solicitante puede ser un proceso, una aplicación, o uno o más procesos hijo de una aplicación. Un recurso puede ser un archivo, un objeto, datos, o cualquier otro tipo de recurso en un ambiente de computación. En las realizaciones, un recurso puede residir en un servidor autónomo o puede residir en un ambiente en clúster. En las realizaciones descritas aquí, un ambiente en clúster puede

incluir uno o más nodos (por ejemplo, dispositivos cliente y/o servidor).

En una realización a modo de ejemplo, una aplicación que reside en un nodo en un ambiente en clúster puede solicitar acceso a un recurso en particular. En las realizaciones, el recurso puede estar almacenado en forma local (por ejemplo, en un nodo cliente), en un dispositivo remoto (por ejemplo, en un servidor remoto o en un nodo diferente en el ambiente de cliente en clúster), o en un ambiente en clúster (por ejemplo, en un ambiente que contiene múltiples nodos) que es diferente del ambiente de cliente en clúster. Por ejemplo, en las realizaciones el ambiente en clúster puede ser un cliente o servidor de clúster, sin embargo, una persona entendida en la materia apreciará que los sistemas y métodos descritos aquí pueden ser empleados en cualquier otro tipo de ambiente, tal como, sin embargo no limitado a, una red virtual.

En tales ambientes, los recursos pueden ser compartidos entre los clientes y las aplicaciones. Cuando una aplicación accede a un recurso, el recurso puede estar enrejado o cerrado, prohibiendo de esta manera que otras aplicaciones accedan al recurso hasta que la aplicación de acceso libere el recurso. El enrejado y cierre del recurso puede ser empleado para protegerse contra un conflicto, es decir, protegerse contra la modificación del recurso por otra aplicación antes de que la aplicación de acceso haya realizado sus operaciones en el recurso. Sin embargo, si el nodo en un ambiente de cliente en clúster falla, la aplicación que está accediendo al recurso puede liberar el recurso de manera inapropiada desde un estado enrejado o cerrado. Por ejemplo, el nodo cliente que accede al recurso por cuenta de la aplicación, puede perder una conexión en red, puede colisionar, o de otro modo puede perder el acceso al recurso antes de que la aplicación

complete sus operaciones y libere en forma apropiada el recurso. Por lo tanto, el recurso puede permanecer en un estado en el cual este no está disponible para otros clientes o aplicaciones. Se pueden emplear mecanismos que liberan en forma automática un recurso desde un estado enrejado o cerrado, previniendo de esta manera que el recurso sea bloqueado en forma permanente. Sin embargo, tales mecanismos frecuentemente esperan un período de tiempo antes de liberar un recurso enrejado o cerrado.

En algunos casos, cuando la aplicación ejecuta un fallo para migrar desde el nodo cliente fallado a un nodo cliente diferente en el cliente de clúster, la aplicación puede intentar restablecer su conexión anterior con el servidor y resumir su(s) operación(es) en el recurso a través del nodo cliente diferente. Sin embargo, debido a que el recurso no fue liberado de manera apropiada por el nodo cliente que ha fallado, el cual anteriormente accedió al recurso por cuenta de la aplicación, debido al error, la aplicación que tuvo acceso anteriormente al recurso puede no ser capaz de resumir su acceso al recurso hasta que el servidor libere el recurso desde su estado enrejado o cerrado. Sin embargo, debido a que el nodo diferente ahora está intentando acceder al recurso por cuenta de la aplicación, el servidor puede no ser capaz de identificar la aplicación como la misma aplicación que estableció anteriormente el cierre en el recurso. Sin embargo, debido a que la misma aplicación está tratando de acceder al recurso, no existe una situación de conflicto. En tales situaciones, la espera para que el servidor libere el cierre anterior en el recurso, puede producir un retardo inaceptable para la aplicación.

Como se describe, debido a que la aplicación está operando en un ambiente de cliente en clúster, cuando la aplicación

solicita el acceso al recurso por una segunda vez, la solicitud para acceder al recurso se puede hacer desde una ubicación diferente, tal como, desde un nodo diferente en el ambiente de cliente en clúster. Por lo tanto, la segunda solicitud puede venir desde una ubicación o dirección IP diferente. Debido a que la solicitud puede ser efectuada desde una ubicación diferente, un servidor puede tener dificultad al asegurar que el cliente o la aplicación intenta acceder nuevamente al recurso, realmente es el mismo cliente que accedió anteriormente al recurso. Los sistemas y métodos descritos aquí proveen mecanismos para identificar situaciones en donde la misma aplicación intenta acceder a un recurso, evitando de ese modo tal retardo y, suministrando un acceso continuo de aplicación al recurso.

La Fig. 1 ilustra un sistema 100 que puede ser usado para implementar algunas de las realizaciones descritas aquí. El sistema 100 incluye un cliente de clúster 102 y un servidor de clúster 106. El cliente de clúster incluye múltiples nodos, tales como los clientes 102A y 102B. Los clientes 102A y 102B pueden ser un dispositivo o aplicación que reside en el cliente de clúster 102. El cliente de clúster 102 puede comunicarse con el servidor de clúster 106 a través de la red 108. En las realizaciones, la red 108 puede ser la Internet, una WAN, una LAN, o cualquier otro tipo de red conocida en la materia. El servidor de clúster 106 almacena recursos a los cuales pueden acceder las aplicaciones en el cliente de clúster 102 (por ejemplo, las aplicaciones que residen en el Cliente 102A o Cliente 102B). En las realizaciones, un cliente (por ejemplo, el Cliente 102A) puede establecer una sesión con el clúster 106 para acceder a los recursos en el clúster 106 por cuenta de una aplicación que reside en el cliente. Aunque en la Fig. 1 el cliente de clúster 102 solamente incluye dos

clientes (por ejemplo, el Cliente 102A y el Cliente 102B), una persona entendida en la materia apreciará que cualquier cantidad de clientes pueden ser incluidos en el cliente de clúster 102.

Como se muestra en la Fig. 1, el servidor de clúster 106 incluye los servidores 106A, 106B, y 106C, los que proveen tanto la disponibilidad como la redundancia para la información almacenada en el clúster 106. En las realizaciones, el clúster 106 puede tener un sistema de archivos, una base de datos, u otra información a la cual puedan acceder los clientes 102 y 104. Aunque se muestran tres servidores en la Fig. 1, en otras realizaciones el clúster 106 puede incluir más de tres servidores, o menos de tres servidores. Más aún, mientras las realizaciones descritas aquí se refieren a un cliente que se comunica con un servidor que es parte de un servidor de clúster, una persona entendida en la materia apreciará que las realizaciones descritas aquí, también pueden ser ejecutadas usando un servidor autónomo.

En las realizaciones, el cliente de clúster 102 provee mecanismos de fallo que le permiten a un cliente migrar desde un primer nodo cliente a un segundo nodo cliente en caso de un error o fallo que ocurra en el primer nodo cliente. Una persona entendida en la materia apreciará que cualquier tipo de mecanismo de fallo puede ser empleado con los sistemas y métodos descritos aquí. Los métodos y sistemas descritos aquí pueden ser empleados para evitar el retardo indebido cuando una aplicación intenta recuperar el acceso a un recurso que migra desde un cliente a otro (por ejemplo, desde el Cliente 102A al Cliente 102B), en el caso de un fallo. En las realizaciones, un identificador de instancia de la aplicación que identifica la aplicación que accede al recurso puede estar asociado con el recurso. El identificador de ejemplo de la

aplicación puede ser un identificador único global (GUID) que está asociado con una aplicación, una sección ejecutada por una aplicación, o un proceso hijo de una aplicación. Por ejemplo, en una realización, una aplicación puede estar asociada con un identificador de instancia de la aplicación que es un GUID. En otra realización, un identificador de instancia de la aplicación puede estar asociado con una operación o acción específica ejecutada por una aplicación. Por ejemplo, si la aplicación emite dos solicitudes abiertas diferentes para dos archivos diferentes, cada solicitud abierta puede tener su propio identificador de instancia de la aplicación. En aún otra realización, un identificador de instancia de la aplicación puede estar asociado con uno o más procesos hijo de la aplicación. Como será claro para aquellas personas entendidas en la materia a partir de las realizaciones descritas aquí, asociar el identificador de instancia de la aplicación de una aplicación con su uno o más procesos hijo, permitirá que los procesos hijo accedan al recurso, si el recurso está ubicado en un estado cerrado o enrejado que pertenece a la aplicación. En las realizaciones, el identificador de instancia de la aplicación puede ser enviado por el cliente cuando se envía o, después de enviar una solicitud para un recurso.

De acuerdo con otra realización, además de almacenar información a la cual acceden los clientes que son una parte del cliente de clúster 102, el servidor de clúster 106 además provee un mecanismo de fallo que permite el acceso continuo de un recurso, en caso de fallo de un nodo servidor. Nuevamente, una persona entendida en la materia apreciará que cualquier tipo de mecanismo de fallo puede ser empleado con los sistemas y métodos descritos aquí.

En las realizaciones, cuando un cliente solicita el acceso a un recurso por cuenta de una aplicación, el identificador de instancia de la aplicación es enviado con la solicitud. El servidor que recibe la solicitud puede asociar el identificador de instancia de la aplicación con el recurso. Por ejemplo, el servidor de clúster puede almacenar el identificador de instancia de la aplicación en una tabla o caché ubicados en uno o más nodos (por ejemplo, servidores tales como los servidores 106A, 106B, y/o 106C) ubicados en el servidor de clúster 106, de tal modo que el identificador de instancia de la aplicación esté asociado con el recurso. Antes de que el cliente haya terminado con el recurso, el cliente puede experimentar un error que lo forzará a perder la conexión con el recurso. Por ejemplo, el cliente que aloja la aplicación o ejecuta solicitudes u operaciones por cuenta de la aplicación, puede perder su conexión de red al servidor de clúster, el cliente puede colisionar, o puede ocurrir cualquier otro tipo de error que interfiera con las aplicaciones que usan el recurso. Al experimentar el error, la aplicación puede producir un fallo a un nuevo nodo cliente en el cliente de clúster 102. El nuevo nodo cliente puede reconectarse con el servidor de clúster y enviar una segunda solicitud para acceder al recurso por cuenta de la aplicación. En las realizaciones, el cliente puede reconectarse al mismo nodo en el servidor de clúster 106 o a un nodo diferente. La segunda solicitud para acceder al recurso puede incluir el identificador de instancia de la aplicación, de la aplicación. Al recibir la segunda solicitud, el servidor (por ejemplo, un Servidor 106A del servidor de clúster 106) compara el identificador de instancia de la aplicación de la segunda solicitud con el identificador de instancia de la aplicación asociado con el recurso. Si los dos identificadores de

instancia de la aplicación coinciden, el servidor de clúster invalida el recurso. En las realizaciones, la invalidación del recurso puede comprender cerrar un archivo, remover un cierre en el recurso, o de otro modo tomar cualquier acción que libere el recurso para su uso. El nodo servidor entonces puede garantizar la segunda solicitud de la aplicación para acceder al recurso. Si el identificador de instancia de la aplicación del segundo nodo no coincide con el identificador de aplicación asociado con el recurso, el servidor no permitirá el acceso al recurso hasta que el recurso esté libre.

Para ilustrar una realización, un solicitante (por ejemplo, un proceso, una aplicación, etc.) en el Cliente 102A en el cliente de clúster 106, puede solicitar que el Cliente 102A establezca una sesión con un servidor del servidor de clúster 106. Por ejemplo, el Cliente 102A puede establecer una sesión con el servidor 106A para acceder a una base de datos almacenada que está en el servidor 106A o que es una parte del servidor de clúster 106, en donde el servidor 106A puede acceder a la base de datos. El cliente 102A entonces envía una solicitud para un recurso por cuenta del solicitante. Un identificador de instancia de la aplicación que identifica al solicitante, está asociado con la solicitud. En las realizaciones, la solicitud puede incluir el identificador de instancia de la aplicación o el identificador de instancia de la aplicación puede ser enviado en forma separada en una forma tal, que el servidor 106A pueda determinar que el identificador de instancia de la aplicación está asociado con la solicitud. En aún otra realización, el servidor 106A o el servidor de clúster 106A pueden ya tener la información necesaria para asociar el identificador de instancia de la aplicación con la solicitud, sin tener que recibir al identificador de instancia de la aplicación junto con la

solicitud. El servidor 106A entonces le garantiza al solicitante el acceso al recurso, permitiendo de esta manera que el solicitante ejecute las operaciones o, de otro modo, acceda al recurso. Cuando se le garantiza al solicitante el acceso al recurso, el servidor 106A asocia un identificador de instancia de la aplicación con el recurso en una forma que indica que el solicitante está accediendo en ese momento al recurso. El recurso entonces puede ser enrejado o cerrado, de modo tal que otros clientes o aplicaciones no puedan acceder o modificar el recurso hasta que el cliente 102 haya completado su operación.

Antes que el solicitante complete sus operaciones en el recurso, ocurre un error que hace que el Cliente 102A falle o, que de otro modo, pierda su conexión al recurso. Debido a que el solicitante cliente completó su operación, este no ha liberado el control del recurso. Por lo tanto, el recurso puede permanecer en un estado enrejado o cerrado. El solicitante o cliente de clúster 102 puede emplear un mecanismo de fallo para migrar al solicitante desde el cliente 102A al cliente 102B. Una vez que la operación de fallo se completa, el cliente 102B puede reconectarse al servidor de clúster 106 por cuenta del solicitante. El cliente 102B puede reconectarse al servidor 106A o establecer una nueva conexión con cualquier otro servidor en el servidor de clúster 106 (por ejemplo, el servidor 106B o 106C). En una situación a modo de ejemplo, el Cliente 102B se reconecta al servidor 106A. Al reconectarse, el cliente 102B puede enviar una segunda solicitud para acceder al recurso por cuenta del solicitante. Como se indicara anteriormente, debido a que el solicitante no liberó el control del recurso, el recurso aún puede estar en un estado cerrado o enrejado. De modo tal de acceder al recurso, sin tener que esperar que el servidor cambie de

manera automática el estado del recurso, por ejemplo, a través de una operación de tiempo muerto, el solicitante puede nuevamente proveer su identificador de instancia de la aplicación con la segunda solicitud. El servidor 106A compara el identificador de instancia de la aplicación provisto, con la segunda solicitud al identificador de instancia de la aplicación asociado con el recurso. Por ejemplo, al comparar el identificador de instancia de la aplicación recibido o, de otro modo, asociado con la segunda solicitud a un identificador de instancia de la aplicación, que el Servidor 106A asoció con el recurso. El identificador de instancia de la aplicación asociado puede estar almacenado en un caché o tabla local del servidor 106A, o puede estar almacenado en otras partes en el servidor de clúster 106. Si el identificador de instancia de la aplicación almacenado en el caché coincide con el identificador de instancia de la aplicación que está asociado con el recurso, el servidor 106A invalida o, de otro modo libera el recurso y le permite al cliente 102B acceder nuevamente al recurso por cuenta del solicitante, sin tener que esperar que el recurso sea liberado por medio de algún otro mecanismo (por ejemplo, por el tiempo muerto del estado enrejado o cerrado). Si los identificadores de instancia de la aplicación no coinciden, el Cliente 102B tendrá que esperar a que el recurso se libere, antes de acceder a este.

En tanto en el ejemplo anterior el Cliente 102B se reconectó al mismo servidor 106A, también es posible en otras realizaciones, que el cliente se conecte a otro nodo en el servidor de clúster 106. Por ejemplo, el cliente 102B puede reconectarse al servidor 106B y presentar una segunda solicitud para recuperar el acceso al recurso por cuenta del solicitante. La segunda solicitud nuevamente puede estar

asociada con el identificador de instancia de la aplicación del solicitante, por ejemplo, al ser incluido en la segunda solicitud o, de otro modo, asociado con la segunda solicitud. En este ejemplo, el Servidor 106B puede no tener el identificador de instancia de la aplicación asociado con el recurso almacenado en su caché local, debido a que el acceso original del recurso estaba en el servidor 106A. En tal situación, el servidor 106B puede contactar a los otros servidores en el servidor de clúster 106 para determinar si estos tienen un identificador de instancia de la aplicación asociado con el recurso. Si el identificador de aplicación asociado con el recurso es almacenado en un nodo diferente en el servidor de clúster (por ejemplo, el servidor 106A), el identificador de instancia de la aplicación en el otro nodo en el servidor de clúster es comparado con el identificador de instancia de la aplicación provisto con la segunda solicitud. Si estos coinciden, el servidor 106B puede enviarle una solicitud al servidor 106A para invalidar el recurso, y entonces el servidor 106B puede permitirle al solicitante (ahora en el cliente 102B), acceder al recurso. Si los identificadores de instancia de la aplicación no coinciden, el Cliente 102B tendrá que esperar para que el recurso se libere.

En base a los ejemplos anteriores, una persona entendida en la materia apreciará que cualquier nodo cliente en el cliente de clúster 102 puede solicitar el acceso, y entonces proveerle el acceso a un solicitante en el cliente de clúster 102. Más aún, cualquier nodo servidor en un servidor de clúster (por ejemplo, cualquier servidor en el servidor de clúster 106) es capaz de determinar si el solicitante tuvo acceso anteriormente al recurso, incluso si el acceso ocurrió en un nodo servidor diferente en el servidor de clúster. Una persona entendida en la materia apreciará que la siguiente

descripción es simplemente un ejemplo de cómo la realización mostrada en la Fig. 1 puede operar, y existen otras realizaciones. Por ejemplo, más que acceder a recursos en un servidor remoto o servidor de clúster, un nodo cliente puede ejecutar las realizaciones descritas aquí para proveerle a los solicitantes (por ejemplo, aplicaciones o procesos) el acceso continuo a los recursos que residen en el ambiente en clúster (por ejemplo, en los mismos nodos de cliente de clúster o diferentes que componen el cliente de clúster). Como se describe con más detalles más adelante, las realizaciones descritas aquí pueden involucrar varios pasos u operaciones diferentes. Más aún, las realizaciones descritas aquí pueden ser implementadas usando cualquier módulo o componente de software o de hardware apropiados.

Yendo ahora a la Figura 2, la figura ilustra un diagrama de bloques de un ambiente de software 200 que muestra el nodo cliente de clúster 201 con múltiples nodos cliente (por ejemplo, los clientes 202 y 204) y un nodo de servidor de clúster 206 con múltiples nodos servidores (por ejemplo, el Nodo 1 208 y el Nodo 2 216). En las realizaciones, el cliente 202 solicita el acceso a un recurso, tal como el recurso 226, en un ambiente de servidor de clúster 206 por cuenta de un solicitante. El nodo cliente de clúster 201 puede ser un cliente de clúster tal como un cliente de clúster 102 (Fig. 1). Aunque no se ilustra, el cliente de clúster puede contener más de dos clientes. El nodo de servidor de clúster 206 puede ser un servidor de clúster, tal como el servidor de clúster 106 (Fig. 1), o puede ser cualquier otro tipo de ambiente de clúster tal como, sin embargo no limitado a, una red virtual. El recurso 226 puede ser almacenado en un almacén de datos 228 que es parte del ambiente en clúster. Aunque no se muestra, en las realizaciones alternativas, el almacén de datos 228 puede

no ser parte del ambiente en clúster, sin embargo, puede estar conectado al ambiente en clúster sobre una red. Los ejemplos de tal red incluyen, sin embargo no están limitados a, la Internet, a una WAN, a una LAN, o a cualquier otro tipo de red conocida en la materia. En aún otras realizaciones, el almacén de datos puede ser parte de un nodo (por ejemplo, un dispositivo) que es una parte del clúster 206.

El nodo servidor de clúster 206 puede incluir uno o más nodos, tales como el Nodo 1 208 y el Nodo 2 216. Aunque se ilustran solamente dos nodos de clúster en la Fig. 2, se pueden incluir cualquier cantidad de nodos de clúster en el ambiente de clúster 206. En las realizaciones, los nodos de clúster 208 y 216 son capaces de recibir una solicitud para ejecutar una operación, y/o garantizar el acceso al recurso 226. En las realizaciones, el recurso 226 puede ser un archivo, un objeto, una aplicación, datos, o cualquier otro tipo de recurso almacenado en, o asequible por el nodo en el nodo de clúster 206 o, por un servidor autónomo.

En las realizaciones, un cliente le envía una solicitud inicial 222 al ambiente de clúster 206. Como se ilustra en la Fig. 2, la solicitud inicial 222 puede ser enviada por el cliente 202 y ser recibida por el Nodo 1 208. Sin embargo, en las realizaciones alternativas, la solicitud inicial 222 puede ser enviada por el cliente o por cualquier otro nodo cliente en el cliente de clúster 201 y, ser recibida por el Nodo 2 216 o por cualquier otro nodo en el servidor de clúster 206. Un ejemplo de solicitudes incluye, sin embargo no está limitado a, solicitudes para crear, abrir, o de otro modo acceder a un archivo. La solicitud 222 puede ser transmitida desde el cliente al Nodo de clúster a través de una red tal como, sin embargo no limitada a, la Internet, una WAN, una LAN, o cualquier otro tipo de red conocida en la materia. La

solicitud inicial 222 puede incluir una solicitud para acceder a un recurso, tal como al recurso 226. En las realizaciones, la solicitud 222 también puede incluir un identificador de instancia de la aplicación que le identifique al solicitante que el cliente 202 está haciendo la solicitud. En las realizaciones, la solicitud inicial 222 puede consistir de uno o más mensajes. Por ejemplo, la solicitud 222 puede ser un solo mensaje que contiene tanto la solicitud como un identificador de instancia de la aplicación. En otra realización, la solicitud 222 pueden ser múltiples mensajes que incluyan una o más solicitudes, así como uno o más identificadores de instancia de la aplicación. En las realizaciones, el cliente 202 puede incluir una aplicación de instancia de caché 214 que se usa para almacenar y/o para generar uno o más identificadores de instancia de la aplicación que pueden ser transmitidos con la solicitud 222.

Como se muestra en la Fig. 2, el Nodo 1 208 puede recibir la solicitud 222 y un identificador de instancia de la aplicación desde el cliente 202. Si el recurso 226 solicitado está disponible, por ejemplo, no enrejado o cerrado por otro cliente o aplicación, el Nodo 1 puede garantizar la solicitud del cliente (por ejemplo, el cliente 202) para acceder al recurso 226 por cuenta de un solicitante que se está ejecutando en el cliente. Al garantizar el acceso al recurso 226, el controlador del filtro 210 puede asignar o, de otro modo, crear una asociación entre el cliente 202 y el recurso 226, al almacenar el identificador de instancia de la aplicación que recibió del cliente 202. En las realizaciones, la asociación puede ser almacenada como un objeto en la aplicación de instancia de caché 212 que es una parte del Nodo 1. Aunque la realización ilustrada muestra la aplicación de instancia de caché 212 como una parte del Nodo 1 208, en las

realizaciones, la aplicación de instancia de caché 212 puede ser almacenada en otras partes, como una parte del nodo de clúster 206. Una persona entendida en la materia apreciará que el Nodo de clúster 206 puede incluir una o más aplicaciones de instancia de caché, tal como una aplicación de instancia de caché 220 en el Nodo 2 216. En las realizaciones, cuando más de una aplicación de instancia de caché está presente, los datos almacenados en la aplicación de instancia de caché múltiple pueden ser repetidos a través de todas las aplicaciones de instancia de caché, o cada aplicación de instancia de caché puede almacenar datos separados.

En una realización, el identificador de instancia de la aplicación recibió del cliente un identificador de instancia de la aplicación que identifica a un solicitante (por ejemplo, una aplicación o proceso) que puede estar almacenado en una estructura `_NETWORK_APP_INSTANCE_ECP_CONTEXT`. La estructura `_NETWORK_APP_INSTANCE_ECP_CONTEXT` puede ser definida como es indica a continuación:

```

    Typdef struct _NETWORK_APP_INSTANCE_ECP_CONTEXT{
        USHORT Size;

        USHORT Reserved;

        GUID AppInstanceID;

    } _NETWORK_APP_INSTANCE_ECP_CONTEXT,
*PNETWORK_APP_INSTANCE_ECP_CONTEXT;

```

En tales realizaciones, el tamaño variable puede almacenar información relacionada con el tamaño de la estructura y la variable `AppInstanceID` puede ser un identificador de instancia de la aplicación único para una aplicación de cliente de clúster con fallo, tal como un solicitante que se ejecuta en el cliente 202. En las realizaciones, la `_NETWORK_APP_INSTANCE_ECP_CONTEXT`, u otro objeto o variable que contenga el identificador de instancia de la aplicación

del solicitante pueden ser almacenados en el caché del identificador único global (GUID) 214. En las realizaciones, la estructura `_NETWORK_APP_INSTANCE_ECP_CONTEXT` puede ser enviada desde un cliente a un servidor en asociación con una solicitud para acceder a un recurso (por ejemplo, una solicitud de creación o abierta). En una realización, el identificador de instancia de la aplicación del solicitante puede ser almacenado en el caché del GUID del nodo cliente que el solicitante está ejecutando en el ambiente de cliente en clúster 201. En otra realización, aunque no se muestra en la Fig. 2, el nodo cliente de clúster 201 puede tener un repositorio central que almacena los identificadores de instancia de la aplicación. En tal realización, múltiples nodos cliente en el nodo cliente de clúster 201 pueden acceder al repositorio centralizado. En aún otra realización, los identificadores de instancia de la aplicación pueden ser almacenados a través de múltiples cachés del GUID (por ejemplo, caché del GUID 214 y caché del GUID 216). En tales realizaciones, el nodo cliente de clúster 201 puede emplear un algoritmo de replicación para asegurar que los múltiples cachés del GUID contienen los mismos identificadores de instancia de la aplicación.

Como se describiera anteriormente, el identificador de instancia de la aplicación puede estar asociado con el recurso 226, mientras el cliente 202 accede al recurso 226 por cuenta de un solicitante. Un nodo servidor 206 puede almacenar tal asociación en uno o más cachés de instancia de la aplicación que son parte del nodo servidor de clúster 206, tal como los cachés de instancia de la aplicación 212 y 220. En una realización, el identificador de instancia de la aplicación puede estar asociado con el recurso al agregarlo a una lista del Parámetro de Creación Extra (ECP) para el recurso 226. La

lista del ECP puede ser almacenada en un caché de instancia de la aplicación que es parte del nodo servidor de clúster 206, tal como los cachés de instancia de la aplicación 212 y 220. En las realizaciones, cuando un ECP es recibido por un servidor, el servidor extrae un identificador de instancia de la aplicación desde el ECP y lo agrega a un caché para asociarse con un recurso, manejar un recurso, etc. Como se describe con respecto al almacenamiento de los identificadores de instancia de la aplicación en el cliente de clúster 201, los identificadores de instancia de la aplicación asociados con un nodo pueden ser almacenados en un caché de instancia de la aplicación individual en el nodo servidor de clúster 206, en un repositorio central en el servidor de clúster 206, o replicado a través de múltiples cachés de instancia de la aplicación en múltiples nodos en el nodo servidor de clúster 206.

En las realizaciones, el recurso 226 está enrejado o cerrado en tanto que un solicitante que se ejecuta en el cliente 202 tiene acceso al recurso 222, previniendo de esta manera que otro cliente o aplicaciones tengan acceso al recurso 226 y evitando cualesquiera conflictos potenciales. En las realizaciones, antes de que el solicitante complete su operación en el recurso 226, el cliente 202 experimenta un error que hace que este pierda la conexión con el recurso. Por ejemplo, el cliente puede colisionar, puede desconectarse, o perder su conexión de red al nodo servidor 208. En tales casos, el recurso 226 aún puede estar en un estado enrejado o cerrado debido a que el solicitante no liberó un cierre en el recurso, previniendo de esta forma que otros clientes accedan al recurso 226.

Cuando le ocurre el error al cliente 202, el solicitante puede usar un mecanismo de fallo de cliente 232 para migrar a

un nuevo nodo cliente (por ejemplo, el cliente 204) en el cliente de clúster 201. Una persona entendida en la materia apreciará que cualquier tipo de mecanismo de fallo puede ser empleado en el fallo de cliente 232. En las realizaciones, el mecanismo de fallo 232 también puede incluir la migración del identificador de instancia de la aplicación del solicitante, el cual puede haber sido almacenado en el caché del GUID 214 en el cliente 202, que ahora ha fallado. Al completar la migración, el solicitante puede intentar recuperar el acceso al recurso 202. En las realizaciones, el cliente 216 puede enviar una segunda solicitud 224 al Nodo 1 para solicitar el acceso al recurso 226 por cuenta del solicitante. Sin embargo, sin las realizaciones de acceso continuo descritas aquí, cuando el Nodo 1 208 recibe una solicitud para acceder al recurso 226 por cuenta del cliente 204 (el remitente de la segunda solicitud 224), este puede negar la solicitud debido a que el recurso 226 está aún en un estado enrejado o cerrado desde el acceso anterior que hizo el cliente 202 por cuenta del recurso. Sin las realizaciones descritas aquí, el Nodo 1 208 podría reconocer que la segunda solicitud para acceder al recurso 226 fue desde una ubicación diferente (por ejemplo, el cliente 204). El Nodo 1 208 podría no ser capaz de determinar que la solicitud es para el mismo solicitante que sostiene el cierre en el recurso 226, y por lo tanto podría determinar que el garantizar la solicitud, podría resultar en un conflicto. Sin embargo, si el mismo solicitante intenta acceder al recurso 224, no existe asunto de conflicto y, forzar al cliente para que espere a que el recurso sea liberado por el sistema, puede resultar en retardos indebidos.

El identificador de instancia de la aplicación puede ser usado para resolver este problema. En las realizaciones, la segunda solicitud 224 además puede incluir que el

identificador de instancia de la aplicación identifique al solicitante que migró al cliente 204 durante el fallo mostrado en 232. En las realizaciones, el identificador de instancia de la aplicación del solicitante puede estar presente en el caché del GUID 228 del cliente 204 antes de la migración del solicitante durante el fallo de cliente 232. Por ejemplo, un mecanismo de replicación puede haber sido empleado para replicar el identificador de instancia de la aplicación del solicitante a través de los nodos en el cliente de clúster 201. En otra realización, el solicitante 203 puede almacenar su identificador de instancia de la aplicación. En aún otra realización, el identificador de instancia de la aplicación del solicitante 203 puede ser migrado durante el fallo de cliente 232.

Como se describe con respecto a la solicitud 222, el identificador de instancia de la aplicación puede ser transmitido en el mismo mensaje que la segunda solicitud 224 o, la segunda solicitud 224 puede estar compuesta de una cantidad de diferentes mensajes. Cuando la segunda solicitud es recibida en el nodo de clúster 206, o un nodo individual en el clúster, tal como el Nodo 1 208, y el servidor de recepción determina que el recurso está enrejado o cerrado, se hace una determinación de si el identificador de instancia de la aplicación en la segunda solicitud 224 es la misma que el identificador de instancia de la aplicación asociado con el recurso 226. En las realizaciones, el Nodo 2 216 comparará el identificador de instancia de la aplicación recibido con la segunda solicitud 222 con el identificador de instancia de la aplicación que está asociado con el recurso 226. El identificador de la aplicación asociado con el recurso 220 puede estar almacenado en el caché de instancia de la aplicación 212 del Nodo 1 212. En las realizaciones en donde

múltiples cachés de instancia de la aplicación existen en el nodo de clúster 206, la determinación puede verificar más de un caché de instancia de la aplicación en el nodo de clúster 206. En tales realizaciones, si un identificador de instancia de la aplicación coincidente no está ubicado en el caché de instancia de la aplicación 212, el Nodo 1 216 puede enviar una solicitud al Nodo 2 212 para determinar si un identificador de instancia de la aplicación coincidente está ubicado en el caché de instancia de la aplicación 220.

En una realización, si el identificador de instancia de la aplicación recibido en la segunda solicitud 224 no coincide con el identificador de instancia de la aplicación asociado con el recurso 226 (el cual puede estar almacenado en el caché de instancia de la aplicación 212 y/o 220), la segunda solicitud 224 puede estar garantizada hasta que el recurso 226 esté liberado. Sin embargo, si se encuentra una coincidencia, el servidor de recepción (por ejemplo, el Nodo 1 208) y/o el nodo servidor de clúster 206 ejecutan acciones para garantizar el acceso al recurso 226 sin causarles un retardo indebido al cliente 204 y al solicitante 203. En tales casos, el nodo de clúster 206 puede invalidar el recurso 226, removiendo de esta manera el recurso 226 desde un estado enrejado o cerrado. En las realizaciones, la invalidación de un acceso anterior puede comprender cualquier acción que saque a un recurso de un estado enrejado o cerrado. Un ejemplo no limitante está cerrando un archivo abierto (por ejemplo, si el recurso 226 es un archivo). Una vez que el acceso anterior es invalidado, la segunda solicitud 224 para acceder al recurso 226 puede ser garantizada, suministrando de esta manera el acceso continuo al solicitante 203.

En una realización, el nodo que recibe la segunda solicitud 224, tal como el Nodo 1 208 en la Fig. 2, puede

ejecutar las acciones requeridas para invalidar el acceso anterior del recurso 226, si un nodo diferente (por ejemplo, el Nodo 2 216) tiene acceso y/o permiso para invalidar el acceso anterior. Sin embargo, en algunos casos, el nodo que recibe la solicitud puede no tener acceso o permiso para invalidar el acceso anterior. Por ejemplo, tal instancia puede ocurrir si la solicitud original 222 fue hecha por el Nodo 2 216, en cuyo caso el Nodo 2 216 puede tener control sobre el recurso. En tales casos, el nodo que recibe la segunda solicitud 224 puede enviarle una solicitud al nodo de control para invalidar el acceso anterior. Una vez que el nodo de control ha invalidado el acceso anterior, el nodo que recibe la segunda solicitud 224 puede garantizar la segunda solicitud 224. En otras realizaciones, el nodo que recibe la segunda solicitud 224 puede enviarle una solicitud a un nodo diferente para garantizarle al cliente 204 y/o al solicitante 203 (ahora residiendo en el cliente 204), el acceso al recurso 226.

El proceso descrito evita el retardo indebido al garantizar una segunda solicitud 224 para acceder a un recurso 226 desde un recurso 203 que accedió anteriormente y aún sostiene un cierre en el recurso 226 a través del uso de identificadores de instancia de la aplicación. Más aún, los identificadores de instancia de la aplicación proveen el beneficio de asegurar que cualquier solicitud garantizada no crea un conflicto en el recurso 226. Por ejemplo, si la solicitud fue recibida desde una aplicación diferente, la solicitud incluirá un identificador de instancia de la aplicación que es diferente del identificador de instancia de la aplicación, asociado con el recurso que podría dar como resultado que la solicitud está siendo rechazada. Debido a que los identificadores de instancia de la aplicación son identificadores únicos globales, el identificador de instancia

de la aplicación para las diferentes aplicaciones no será el mismo.

La Fig. 3 es una realización de un método 300 que un solicitante puede emplear para ganar el acceso continuo a un recurso en un ambiente de cliente en clúster. Por ejemplo, un solicitante puede ser un cliente, tal como el cliente 202 (Fig. 2), que emplea el método 300 para acceder a un recurso (por ejemplo, el recurso 226). En las realizaciones, el recurso puede residir en una máquina remota, tal como un servidor. El servidor puede ser un servidor autónomo o parte de un ambiente en clúster, tal como el servidor de clúster 206 (Fig. 2). El flujo comienza en la operación 302 en donde una solicitud para un recurso es enviada a un servidor. En las realizaciones, la solicitud puede ser acceder a un recurso. En las realizaciones, el acceso a un recurso puede comprender abrir un archivo, crear un archivo, o de otro modo, acceder a, o ejecutar una operación en un recurso que puede ser remota a un cliente. En las realizaciones, un solicitante puede operar en un ambiente de cliente en clúster. En tales realizaciones, la solicitud enviada a la operación 302 puede ser enviada desde un primer cliente en el ambiente de cliente en clúster.

El flujo continúa a la operación 304, en donde el identificador de instancia de la aplicación es enviado, por ejemplo, a un servidor (por ejemplo, un servidor autónomo o un nodo en un ambiente en clúster). En una realización, el primer cliente que envió la solicitud también puede enviar el identificador de instancia de la aplicación por cuenta del solicitante. Como se describiera anteriormente, un identificador de instancia de la aplicación es un GUID que identifica al solicitante (por ejemplo, una aplicación, un cliente, o un proceso hijo de una aplicación que solicita el acceso a un recurso). En una realización, el identificador de

instancia de la aplicación puede ser enviado en un mensaje transmitido a través de una red. El identificador de instancia de la aplicación puede ser transmitido en el mismo mensaje que contiene la solicitud en la operación 302 o, este puede ser transmitido en un mensaje diferente. En tales realizaciones, un objeto que contiene el identificador de instancia de la aplicación, tal como, sin embargo no limitado al `_NETWORK_APP_INSTANCE_ECP_CONTEXT` descrito con respecto a la Fig. 2, puede ser enviado en la operación 302.

En una realización, se puede usar una interfaz para enviar el identificador de instancia de la aplicación en la operación 304. La interfaz puede ser una interfaz de nivel de núcleo ubicada en un cliente o, disponible a un cliente que opera en un ambiente de cliente en clúster. En las realizaciones, la interfaz de nivel de núcleo puede ser usada por el solicitante y/o el cliente, para enviar un identificador de instancia de la aplicación a un servidor. El siguiente es un ejemplo no limitante de una interfaz de nivel de núcleo que puede ser empleada en la operación 304 para enviar un identificador de instancia de la aplicación:

```
#if (NTDDI_VERSION >= NTDDI_WIN8)
//
// ECP context for an application to provide its instance ID.
//

typedef struct _NETWORK_APP_INSTANCE_ECP_CONTEXT {
    //
    // This must be set to the size of this structure.
    //

    USHORT Size;

    //

```

```

// This must be set to zero.
//

USHORT Reserved;

//

// The caller places a GUID that should always be unique for a single instance of
// the application.
//

GUID AppInstanceID;

} NETWORK_APP_INSTANCE_ECP_CONTEXT,
*PNETWORK_APP_INSTANCE_ECP_CONTEXT;

//

// The GUID used for the APP_INSTANCE_ECP_CONTEXT structure.
//
// {6AA6BC45-A7EF-4af7-9008-FA462E144D74}
//
DEFINE_GUID(GUID_ECP_NETWORK_APP_INSTANCE, 0x6aa6bc45, 0xa7ef,
0x4af7, 0x90, 0x8, 0xfa, 0x46, 0x2e, 0x14, 0x4d, 0x74);

#endif // NTDDI_Version >= NTDDI_WIN8

```

Aunque se provee una interfaz de nivel de núcleo, una persona entendida en la materia apreciará que otras interfaces de nivel de núcleo pueden ser empleadas en la operación 304 para enviar el identificador de instancia de la aplicación.

En una realización, una interfaz de usuario de aplicación (API) puede ser empleada en la operación 304 para enviar un identificador de instancia de la aplicación. En tal realización, el solicitante y/o el cliente puede enviar un identificador de instancia de la aplicación al hacer una llamada a la API. La API puede estar alojada en el cliente que

ejecuta la operación 304 (por ejemplo, el primer cliente en un servidor de clúster) o, la API puede estar alojada en otro dispositivo y estar asequible para el solicitante u otra aplicación o proceso. El siguiente es un ejemplo no limitante de una API que puede ser empleado en la operación 304 para enviar un identificador de instancia de la aplicación:

```
NTSTATUS RegisterAppInstance(
    _in PGUID AppInstance
):
```

Aunque se provee una API específica, una persona entendida en la materia apreciará que otras APIs pueden ser empleadas en la operación 304. Más aún, aunque la operación 304 es ilustrada como una operación discreta, una persona entendida en la materia apreciará que el envío del identificador de instancia de la aplicación puede ser ejecutado en forma simultánea con el envío de la solicitud en la operación 302.

Cuando el recurso solicitado no está cerrado, la solicitud enviada en la operación 302 es garantizada y el flujo continúa a la operación 306, en donde se accede al recurso. Como se describiera anteriormente, el servidor o dispositivo que controla el recurso puede ubicar el recurso en un estado enrejado o cerrado, en tanto que el solicitante accede al recurso en la operación 306. En algún punto mientras se accede al recurso, ocurre un error, tal como los errores descritos con referencia a la Fig. 2, el cual hace que el cliente falle o, de otro modo, pierda la conexión al recurso. El error puede hacer que el cliente (por ejemplo, el primer cliente en el servidor de clúster) pierda el acceso al recurso antes de que el solicitante complete su uso del recurso. Bajo tales circunstancias, el recurso puede no ser liberado de su estado enrejado o cerrado.

El flujo continúa a la operación 308, en donde se ejecuta la operación de fallo. En las realizaciones, la operación de fallo puede comprender clonar al solicitante y su estado a un cliente diferente en el nodo cliente de clúster (por ejemplo, un segundo cliente). En las realizaciones, el estado del solicitante puede ser clonado en el segundo y el solicitante puede ser ejecutado en el segundo cliente en una forma tal que este pueda resumir la ejecución desde el punto en donde el primer cliente falló. En otra realización, el solicitante puede estar en comunicación con el primer cliente (más que en ejecución) al momento del fallo de los primeros clientes. En tales realizaciones, la operación de fallo puede comprender que el solicitante establece comunicaciones con un segundo cliente en el cliente de clúster.

En las realizaciones, la información de estado, incluyendo sin embargo sin estar limitado al identificador de instancia de la aplicación de los solicitantes, puede ser transferida desde el primer cliente al segundo cliente. En una realización, el primer cliente puede enviar un mensaje incluyendo el identificador de instancia de la aplicación del solicitante y/o la información de estado del solicitante. El identificador de instancia de la aplicación y/o el estado pueden ser enviados durante el proceso de fallo o, en las realizaciones, puede ser enviado antes de que el primer cliente falle, tal como, durante un proceso de replicación que clona información a través de los clientes en un ambiente de cliente en clúster. En otra realización, el identificador de instancia de la aplicación del solicitante y/o la información de estado pueden ser almacenados en una ubicación central o repositorio en la red de cliente en clúster. En tales realizaciones, el proceso de fallo puede proveer el segundo cliente con la ubicación del identificador de instancia de la

aplicación del solicitante y/o la información de estado. En aún otra realización, el solicitante puede mantener su identificador de instancia de la aplicación. En tales realizaciones, la operación de fallo de cliente puede comprender la reubicación hacia, o de otro modo, establecer una conexión entre el solicitante y un segundo cliente.

En las realizaciones, después el flujo de la operación de fallo de cliente continúa a la operación 310. En la operación 310, una segunda solicitud para el mismo recurso es enviada al ambiente en clúster. En las realizaciones, la segunda solicitud es enviada por el segundo cliente en el cliente de clúster por cuenta del solicitante. La segunda solicitud puede ser enviada usando la misma manera que se describiera con respecto al primer recurso en la operación 302. De modo tal de mantener el acceso continuo al recurso y evitar el retardo indebido, el flujo continúa a la operación 312, en donde el identificador de instancia de la aplicación es nuevamente enviado al ambiente de clúster. El identificador de instancia de la aplicación puede ser enviado en la operación 308 de acuerdo con una de las realizaciones descritas con respecto a la operación 304. En las realizaciones, debido a que un cliente diferente (por ejemplo, el segundo cliente) está enviando la segunda solicitud, el servidor que recibe la solicitud puede no ser capaz de identificar la segunda solicitud como perteneciente al mismo solicitante que sostiene un cierre en el recurso (por ejemplo, debido a que la solicitud se hace desde una máquina diferente, desde una dirección diferente, etc.). Sin embargo, al enviar los identificadores de instancia de la aplicación en las operaciones 304 y 308, el servidor será capaz de identificar las solicitudes como pertenecientes al mismo solicitante, y garantizará el acceso continuo al recurso como se describiera

anteriormente con respecto a las Figs. 1 y 2. El flujo continúa a la operación 314 y el solicitante resume el acceso al recurso. En las realizaciones, el segundo cliente puede recibir una respuesta a la segunda solicitud desde el servidor que indica que el servidor garantiza la segunda solicitud. En las realizaciones, al recibir la indicación, el segundo cliente puede acceder al recurso por cuenta del solicitante.

La Fig. 4 es una realización de un método 400 ejecutado por un nodo en un ambiente de servidor en clúster para proveer el acceso continuo a un recurso. Las realizaciones del método 400 pueden ser ejecutadas por un nodo tal como el Nodo 1 208 (Fig. 2) en un ambiente en clúster, tal como un nodo de clúster 206 (Fig. 2). En las realizaciones, el método 400 puede ser ejecutado por un nodo que tiene acceso a un recurso. El flujo comienza en la operación 402, en donde el nodo recibe una solicitud para un recurso. En las realizaciones, un recurso puede ser un archivo, un objeto, un método, datos, o cualquier otro tipo de recurso que esté bajo control de, y/o al cual se pueda acceder por la operación de ejecución del nodo 400. Un identificador de instancia de la aplicación puede ser recibido con la solicitud en la operación 402.

El flujo continúa a la operación de decisión 404, en donde se hace una determinación con respecto a si el recurso está en un estado enrejado o cerrado. Una persona entendida en la materia apreciará que cualquier manera de determinar si un recurso está enrejado o cerrado, se puede emplear en la operación 404. Si el recurso no está en un estado enrejado o cerrado, el flujo se ramifica en NO a la operación 412, en donde la solicitud para el recurso está garantizada. En las realizaciones, garantizar la solicitud puede comprender permitirle al solicitante acceder al recurso, ejecutando una operación en el recurso por cuenta del solicitante, o

permitiendo cualquier tipo de acceso o modificación al recurso. Por ejemplo, garantizar la solicitud en la operación 412 puede incluir abrir un archivo o crear un archivo.

Si el recurso está en un estado enrejado o cerrado, el flujo se ramifica en SI desde la operación 404 a la operación de decisión 406. En la operación de decisión 406, el identificador de instancia de la aplicación recibido con la solicitud en la operación 402 es comparado con un identificador de instancia de la aplicación que está asociado con el recurso. Por ejemplo, como se describe con respecto a la Fig. 2, un nodo puede asociar un identificador de instancia de la aplicación con un recurso cuando un cliente o aplicación accede a un recurso. Como se describiera anteriormente, la asociación del identificador de instancia de la aplicación de un solicitante que accede a un recurso puede estar almacenada en un nodo, por ejemplo, en un caché de instancia de la aplicación, como se describiera en varias realizaciones discutidas en la Fig. 2. En las realizaciones, el identificador de instancia de la aplicación que es provisto en un ECP enviado con una solicitud para un recurso, por ejemplo, en una estructura `_NETWORK_APP_INSTANCE_ECP_CONTEXT`, puede ser agregado a una lista de ECP asociada con el recurso.

En una realización, la asociación del recurso de instancia de la aplicación puede residir en forma local en el nodo que ejecuta el método 400. En tales casos, la comparación puede hacerse en un caché de instancia de la aplicación local, residente en el servidor. Sin embargo, como se discutiera con respecto a la Fig. 2, un ambiente en clúster puede contener una cantidad de cachés de instancia de la aplicación distribuidos a través de diferentes nodos. Más aún, cada uno de los cachés de instancia de la aplicación diferentes puede almacenar datos separados y/o diferentes. El identificador de

la aplicación asociado con el recurso enrejado o cerrado puede ser almacenado en un nodo diferente en el ambiente en clúster. En tales casos, la operación 406 puede incluir enviar una solicitud a un nodo diferente para ejecutar la comparación en la operación 406. La solicitud puede incluir el identificador de instancia de la aplicación recibido en la operación 402.

Si el identificador de instancia de la aplicación recibido no es el mismo que el identificador de instancia de la aplicación asociado con el recurso, el flujo se ramifica en NO a la operación 410. En la operación 410, la solicitud para acceder al recurso recibido en la operación 402 es rechazada. En las realizaciones, la solicitud puede ser rechazada de modo tal de evitar un conflicto de recurso. Debido a que el identificador de la aplicación recibido no es el mismo que el identificador de instancia de la aplicación asociado, la solicitud para acceder al recurso recibido en la operación 402 es desde un solicitante o aplicación diferente. Garantizar una solicitud al cliente o aplicación diferente, como puede ser en este caso, puede causar una situación de conflicto que interferirá con la aplicación que actualmente está accediendo al recurso. Por ejemplo, la aplicación diferente puede modificar el recurso en una forma que modifique o, de otro modo, interfiera con las operaciones ejecutadas en el recurso por el solicitante que actualmente sostiene un cierre en el recurso.

Sin embargo, recibir un identificador de aplicación con la solicitud 402 que es el mismo que el identificador de aplicación asociado con los recursos enrejados o cerrados, indica que un error puede haber ocurrido, el que causó que el solicitante que estaba accediendo al recurso perdiera su acceso al recurso sin liberar en forma apropiada el recurso. Por ejemplo, el solicitante puede operar en un nodo cliente de

clúster. El cliente en particular en donde el solicitante estaba operando, puede haber perdido la conexión al servidor o, de otro modo, falló antes de que el solicitante hubiera completado sus operaciones en el recurso. De modo tal de proveer el acceso continuo al recurso, es decir, para permitirle al solicitante recuperar el acceso al recurso sin experimentar un retardo indebido o inaceptable, el flujo se ramifica en SI, a la operación 408.

En la operación 408, el recurso es invalidado. Como se describiera aquí anteriormente, la invalidación del recurso puede incluir cambiar el estado enrejado del recurso o, de otro modo, remover un cierre en el recurso. Por ejemplo, si el recurso es un archivo, la invalidación del recurso puede incluir cerrar el archivo. Una persona entendida en la materia apreciará que cualquier método de liberación de un recurso enrejado o cerrado puede ser empleado en la operación 408.

Nuevamente con referencia a la Fig. 2, en las realizaciones, el acceso a un recurso puede estar bajo el control de un nodo en el ambiente en clúster diferente del nodo que recibe la solicitud para el acceso a un recurso en la operación 402. Por ejemplo, un manejador al recurso puede residir en un nodo diferente en el ambiente en clúster. En tales realizaciones, la invalidación del recurso puede incluir enviar una solicitud al nodo que controla el acceso al recurso para invalidar el recurso. En respuesta al envío de la solicitud, el nodo remoto puede invalidar el recurso.

Después que el recurso es invalidado, el flujo continúa a la operación 412, en donde la solicitud para acceder al recurso, es garantizada. Garantizar la solicitud puede comprender permitir que el solicitante acceda al recurso, ejecutando una operación en el recurso por cuenta del solicitante, o permitir cualquier tipo de acceso o

modificación al recurso. Por ejemplo, garantizar la solicitud en la operación 412 puede incluir abrir un archivo o crear un archivo. Garantizar tal acceso puede ser ejecutado por el nodo que recibe la solicitud en la operación 402, o por otro nodo en el ambiente en clúster.

Los métodos 300 y 400 son simplemente algunos ejemplos de los flujos operacionales que pueden ser ejecutados de acuerdo con las realizaciones. Las realizaciones no están limitadas a la descripción específica provista anteriormente con respecto a las Figs. 3-6 y pueden incluir operaciones adicionales. Más aún, los pasos operacionales descritos pueden estar combinados en otros pasos y/o vueltos a arreglar. Además, se pueden usar menos pasos, o adicionales, empleados con los métodos descritos en las Figs. 3-4.

La Fig. 5 ilustra un sistema de computador general 500, el cual se puede usar para implementar las realizaciones descritas aquí. El sistema de computador 500 es solamente un ejemplo de un ambiente de computación y no pretende sugerir ninguna limitación con respecto al ámbito de uso o funcionalidad de las arquitecturas de red y computador. De ninguna manera debería el sistema de computador 500 ser interpretado como teniendo alguna dependencia o requerimiento, en relación con cualquiera de uno o combinación de componentes ilustrados en el sistema de computador 500 a modo de ejemplo. En las realizaciones, el sistema 500 puede ser usado como los clientes y/o servidores descritos anteriormente con respecto a las Figs. 1 y 2.

En su configuración más básica, el sistema 500 normalmente incluye por lo menos una unidad de procesamiento 502 y la memoria 504. Dependiendo de la configuración exacta y del tipo de dispositivo de computación, la memoria 504 puede ser volátil (tal como RAM), no volátil (tal como ROM, memoria

flash, etc.), o cualquier combinación. Esta configuración más básica está ilustrada en la Fig. 5 por la línea punteada 506. La memoria de sistema 504 almacena las instrucciones 520, tales como las instrucciones que ejecutan los métodos de disponibilidad continua descritos aquí y los datos 522, tales como los identificadores de instancia de la aplicación que pueden ser almacenados en un sistema de almacenamiento de archivos, con almacenamiento tal como el almacenamiento 508.

El término medio legible por computador como se usa aquí, puede incluir medios de almacenamiento por computador. Los medios de almacenamiento por computador pueden incluir medios volátiles y no volátiles, removibles y no removibles implementados en cualquier método o tecnología para el almacenamiento de información, tales como las instrucciones legibles por computador, las estructuras de datos, los módulos de programa, u otros datos. La memoria de sistema 504, el almacenamiento removible, y el almacenamiento no removible 508, son todos ejemplos de medios de almacenamiento por computador (por ejemplo, almacenamiento de memoria). Los medios de almacenamiento por computador pueden incluir, sin embargo no están limitados a, RAM, ROM, memoria de sólo lectura programable y borrable eléctricamente (EEPROM), memoria flash u otra tecnología de memoria, CD-ROM, discos versátiles digitales (DVD), u otro almacenamiento óptico, cassettes magnéticos, cinta magnética, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnético, u otro medio que pueda ser usado para almacenar información y, a los cuales se pueda acceder por el dispositivo de computación 500. Cualquiera de tales medios de almacenamiento por computador puede ser parte del dispositivo 500. El dispositivo de computación 500 además puede tener un(unos) dispositivo(s) de entrada 514 tales como un teclado, un mouse, un lápiz, un

dispositivo de entrada de sonido, un dispositivo de entrada táctil, etc. El(los) dispositivo(s) de salida 516 tales como una pantalla, parlantes, una impresora, etc. también pueden estar incluidos. Los dispositivos antes mencionados son ejemplos y, se pueden usar otros.

El término medio legible por computador como se usa aquí también puede incluir medios de comunicación. Los medios de comunicación pueden estar incorporados por las instrucciones legibles por computador, las estructuras de datos, los módulos de programa, u otros datos en una señal de datos modulada, tales como una onda transportadora u otro mecanismo de transporte, e incluyen cualesquiera medios de entrega de información. El término "señal de datos modulada" puede describir una señal que tiene una o más características establecidas o cambiadas, de tal manera de codificar información en la señal. A modo de ejemplo y no de limitación, los medios de comunicación pueden incluir medios cableados tales como una conexión de red cableada o cableada directa, y medios inalámbricos tales como acústico, de frecuencia de radio (RF), infrarrojo, u otros medios inalámbricos.

Las realizaciones de la invención pueden ser practicadas a través de un sistema en chip (SOC), en donde cada uno o muchos de los componentes ilustrados en la Fig. 5 pueden estar integrados encima de un solo circuito integrado. Tal dispositivo SOC puede incluir una o más unidades de procesamiento, unidades gráficas, unidades de comunicaciones, unidades de virtualización de sistema y varias funcionalidades de aplicación, todas las cuales están integradas (o "quemadas") encima del substrato de chip, como un solo circuito integrado. Cuando está operando a través de un SOC, la funcionalidad descrita aquí con respecto a suministrar el acceso continuo a un recurso, puede operar a través de la

lógica integrada de aplicación específica con otros componentes del dispositivo/sistema de computación 500 en el único circuito integrado (chip).

Se ha hecho referencia a lo largo de esta especificación a "una realización", queriendo decir que una función, estructura, o característica descrita en particular, está incluídas en, por lo menos, una realización. Por lo tanto, el uso de tales frases puede referirse a más que sólo una realización. Más aún, las funciones, estructuras, o características pueden estar combinadas en cualquier manera adecuada en una o más realizaciones.

Sin embargo, una persona entendida en la materia relevante puede reconocer que las realizaciones pueden ser practicadas sin uno o más de los detalles específicos, o con otros métodos, recursos, materiales, etc. En otros casos, las estructuras, recursos, u operaciones bien conocidos pueden no haber sido mostrados o descritos en detalle simplemente para evitar los aspectos oscuros de las realizaciones.

En tanto que las realizaciones y aplicaciones a modo de ejemplo han sido ilustradas y descritas, se deberá entender que las realizaciones no están limitadas a la configuración precisa y a los recursos descritos anteriormente. Varias modificaciones, cambios, y variaciones aparentes para aquellas personas entendidas en la materia pueden hacerse en la organización, operación, y detalles de los métodos y sistemas descritos aquí, sin salirse del ámbito de las realizaciones reivindicadas.

REIVINDICACIONES

1. Un método para proveer el acceso continuo a un recurso, **CARACTERIZADO** porque el método comprende:

Recibir una primera solicitud de un solicitante para acceder a un recurso, en donde la solicitud es recibida desde un primer cliente;

Asociar un primer identificador de instancia de la aplicación con el primer recurso;

Permitir la primera solicitud de acceso al recurso;

Recibir una segunda solicitud del solicitante para el recurso, después de recuperarse de una falla, en donde la segunda solicitud es recibida desde un segundo cliente diferente del primer cliente;

Recibir un segundo identificador de la aplicación asociado con la segunda solicitud;

Determinar si el primer identificador de la aplicación y el segundo identificador de la aplicación son los mismos; y

Cuando el primer y segundo identificadores de la aplicación son los mismos, ejecutar los pasos que comprenden:

Invalidar la primera solicitud; y

Garantizar que la segunda solicitud acceda al recurso.

2. El método de la reivindicación 1, **CARACTERIZADO** porque el primer identificador de la aplicación está asociado con una instancia de la aplicación de una solicitud abierta.

3. El método de la reivindicación 1, **CARACTERIZADO** porque el primer identificador de la aplicación está asociado con un proceso.

4. El método de la reivindicación 1, **CARACTERIZADO** porque el primer identificador de la aplicación está asociado con, por lo menos, un proceso hijo de una aplicación.

5. El método de la reivindicación 1, **CARACTERIZADO** porque asociar el primer identificador de instancia de la aplicación comprende recibir al primer identificador de instancia de la aplicación en una estructura NETWORK_APP_INSTANCE_ECP_CONTEXT.

6. Un método para suministrar un fallo del cliente en clúster, **CARACTERIZADO** porque el método comprende:

Recibir, en un segundo cliente, un identificador de instancia de la aplicación para un solicitante, en donde el solicitante accediera anteriormente a un recurso que está usando un primer cliente;

Enviar, desde el segundo cliente, una segunda solicitud para acceder al recurso por cuenta del solicitante;

Enviar, desde el segundo cliente, el identificador de instancia de la aplicación, al solicitante;

Recibir una indicación de que un servidor garantizó la segunda solicitud; y

Acceder, por el segundo cliente, al recurso por cuenta del cliente.

7. El método de la reivindicación 6, **CARACTERIZADO** porque el servidor que garantiza la segunda solicitud, garantizó anteriormente una primera solicitud para acceder al recurso desde el primer cliente por cuenta del solicitante.

8. El método de la reivindicación 6, **CARACTERIZADO** porque el segundo cliente envía una segunda solicitud en respuesta a un fallo de cliente.

9. Un sistema para facilitar el fallo del cliente en un ambiente en clúster, **CARACTERIZADO** porque el sistema comprende:

Por lo menos, un servidor que comprende:

Por lo menos, un procesador configurado para ejecutar instrucciones ejecutables por computador;

Por lo menos, un medio de almacenamiento legible por computador, almacenando las instrucciones ejecutables por computador que cuando son ejecutadas por el, por lo menos, un procesador provee:

Recibir una primera solicitud para acceder a un recurso desde un primer cliente por cuenta de un solicitante;

Asociar un primer identificador de instancia de la aplicación con el primer recurso;

Permitirle al solicitante acceder al recurso;

Recibir una segunda solicitud para el recurso desde un segundo cliente, en donde el segundo cliente es diferente del primer cliente;

Recibir un segundo identificador de la aplicación asociado con la segunda solicitud;

Determinar si el primer identificador de la aplicación y el segundo identificador de la aplicación son los mismos;

Cuando el primer y segundo identificadores de la aplicación son los mismos, ejecutar los pasos que comprenden:

Invalidar la primera solicitud; y

Garantizar que la segunda solicitud acceda al recurso.

10. El sistema de la reivindicación 18, **CARACTERIZADO** porque el sistema además comprende:

El primer cliente, comprendiendo:

Por lo menos, un procesador configurado para ejecutar instrucciones ejecutables por computador;

Por lo menos, un medio de almacenamiento legible por computador que almacena las instrucciones ejecutables por computador que cuando es ejecutado por el, por lo menos, un procesador:

Envía la primera solicitud;

Envía el identificador de instancia de la aplicación a un segundo cliente.