METHOD FOR PROCESSING ACQUIRE LOCK REQUEST AND SERVER

TECHNICAL FIELD

The present invention relates to computer technologies, and in particular, to a method for processing a lock request and a server.

BACKGROUND

In a distributed system including multiple hosts, mutually exclusive access to a same resource by multiple nodes at the same time is implemented by using a lock server. When needing to perform some operations on a resource, a host first needs to request lock permission from the lock server. The host can perform a corresponding operation, for example, a read operation or a write operation, on the resource only after the host obtains the lock permission. Therefore, performance, availability, and reliability of the lock server directly affect performance, availability, and reliability of the entire distributed system.

In the distributed system, a host communicates with a node by using an NAS (Network Attached Storage, network attached storage) network. A lock server is disposed in each node, each node is further connected to a storage system, and a resource, for example, a file, is stored in the storage system.

When needing to perform an operation (for example, a read operation or a write operation) on the resource in the storage system, the host first applies to a lock server for lock permission by using an application program on the host, and then performs the operation on the file after obtaining the lock permission assigned by the lock server for the resource. A correspondence between lock permission that has been assigned for a resource and an application program may be stored in each node, or may be stored in a shared storage that can be accessed by each node. For example, when needing to perform a read operation on a file in the storage system, the host first applies to a lock server in a node for lock permission for the file, and can perform a read operation on the file only after the host obtains the lock permission for the file. A correspondence between lock permission for a file and an application having the lock permission is stored in a node, and a node or an application on a node has lock permission. Even if a node has lock permission, the node may still learn, by means of a

further analysis, which application on the node needs to use a resource in the storage system.

When a fault occurs in a lock server, a service on the faulty lock server needs to be switched to a lock server in which no fault occurs (which is referred to as a non-faulty lock server below). When a protocol, such as NFS (Network File System, Network File System) or Samba, is used, and when the service on the faulty lock server is switched to the non-faulty lock server, a host may reapply, by using a reclaim lock request, for lock permission that has been obtained for a file by each application, to increase access efficiency of the host. In this way, in a distributed lock server cluster, a reclaim lock request and an acquire lock request need to be securely controlled, to avoid a problem that data viewed by multiple applications is inconsistent because lock permission is inappropriately controlled, and that even data corruption is caused when multiple applications read and write data at the same time.

In the prior art, when a fault occurs in a lock server, all remaining lock servers in the distributed system enter a silent state. During silence, all the lock servers in the distributed system can process only a reclaim lock request for reapplying for lock permission, and cannot process an acquire lock request for applying for new lock permission. Only after processing on a reclaim lock request for lock permission that has been granted by the faulty lock server is completed, the lock servers in the distributed system exit the silent state, and normally process an acquire lock request.

In the foregoing prior-art solution, when a fault occurs in a lock server in the distributed system, all remaining lock servers in the distributed system need to enter a silent state. In this case, the lock servers refuse to process an acquire lock request sent by any lock server, resulting in a noticeable deterioration in performance and reliability of the system.

SUMMARY

According to a first aspect of the present invention, a method for processing a lock request is provided, and can be applied to a first lock server, where the first lock server is a takeover lock server of a second lock server, the first lock server stores a lock management range of the second lock server, and the method includes: entering, by the first lock server, a silent state after learning that a fault occurs in the second lock server, where a silent range of the silent state is a resource for which the second lock server has assigned permission; receiving, by the first lock server, a first acquire lock request, where the first acquire lock request is used to request to lock a first resource, and the first acquire lock request carries an identifier of the first resource; detecting, by the first lock server, that the first resource belongs to the management range of the second lock server; and querying, by the first lock server, a first resource information record list, where the first resource

feedback message to the first lock server, where the feedback message carries lock permission that is obtained through application by the lock proxy by using the second lock server, and the first lock server records the lock permission into a detailed resource information record list.

The present invention further provides implementation manners of a lock request management apparatus and a server, and the lock request management apparatus and the server have functions in the first aspect and the possible implementation manners.

Correspondingly, the present invention further provides a non-volatile computer readable storage medium and a computer program product. When computer instructions included in the non-volatile computer readable storage medium and the computer program product are loaded to a memory of a storage device provided in the present invention, and a central processing unit (Central Processing Unit, CPU) executes the computer instructions, the storage device is enabled to separately perform possible implementation solutions in the first aspect and the possible implementation manners, and the solutions may run in an apparatus or a server for execution.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 is a topology diagram of a use environment of a lock management system according to an embodiment of the present invention;

FIG. 2 is a schematic diagram of an embodiment of a lock server management range and a lock server takeover relationship according to the present invention;

FIG. 3 is a flowchart of a method for processing a lock request according to an embodiment of the present invention;

FIG. 4 is a structural diagram of an embodiment of a lock request management apparatus according to the present invention; and

FIG. 5 is a structural diagram of an embodiment of a server according to the present invention.

DESCRIPTION OF EMBODIMENTS

The following clearly and describes the technical solutions in the embodiments of the present invention with reference to the accompanying drawings in the embodiments of the present invention. Apparently, the described embodiments are only some embodiments rather than all the embodiments of the present invention.

In the embodiments of the present invention, establishing a takeover relationship between lock servers is proposed. In this way, when a fault occurs in a lock server, a takeover lock server of the faulty lock server may be got according to the takeover relationship.

consistent hashing ring is 0–232, results obtained by performing hash calculation on the IDs of the lock servers are sequentially hash(1)=5000, hash(2)=8000, hash(3)=1024, and hash(4)=512. In a clockwise direction, starting from 0, a sequence of locations of the lock servers on the hashing ring is: the lock server 4, the lock server 3, the lock server 1, and the lock server 2. In this case, a management range of the lock server 4 is (8000,232] and [0,512], a management range of the lock server 3 is (512, 1024], a management range of the lock server 1 is (1024, 5000], and a management range of the lock server 2 is (5000, 8000]. In this case, in the ring logically formed by the lock servers in FIG. 2, the takeover relationship between the lock servers is determined in a clockwise direction of the consistent hashing ring, that is, a takeover lock server of the lock server 1 is the lock server 2, a takeover lock server of the lock server 2 is the lock server 4, a takeover lock server of the lock server 4 is 3, and a takeover lock server of the lock server 3 is 1.

Certainly, this embodiment of the present invention provides a method for determining a takeover server of a lock server. It should be noted that a takeover relationship is not unique, as long as each lock server can have a takeover server. For example, a manager may configure a takeover server for each lock server. For example, the lock server 2 is configured as the takeover lock server of the lock server 1, the lock server 3 is configured as the takeover lock server of the lock server 2, the lock server 4 is configured as the takeover lock server of the lock server 3, and the lock server 1 is configured as the takeover lock server of the lock server 4.

One lock server may take over from multiple lock servers. For example, if a fault occurs in both the lock server 3 and the lock server 1, a takeover lock server of the lock server 3 and the lock server 1 is the lock server 4.

In this embodiment of the present invention, after receiving a lock request (for example, a reclaim lock request or an acquire lock request), a lock proxy determines, according to a stored lock server management range, a lock server that should process the lock request. When determining that a fault occurs in the lock server that should process the lock request (when a fault occurs in the lock server, the management node broadcasts a notification message to the lock proxies in the distributed system), the lock proxy determines a takeover lock server according to a lock server takeover relationship, and sends the lock request to the takeover lock server for processing.

The lock server management range and the lock server takeover relationship may be both configured by the management node, and sent to all the lock proxies for storage; or may be sent to the lock proxies after the management node obtains a consistent hashing ring by means of calculation. Alternatively, the management node may configure the lock proxies in advance, and the lock proxies separately perform calculation to obtain a same consistent hashing ring.

After receiving the lock request, the lock proxy performs, by using the consistent hashing

takeover relationship, a takeover lock server of the faulty lock server 2 is the lock server 4. Therefore, the lock proxy sends the reclaim lock request to the takeover lock server 4 for processing. The lock server 4 performs hash calculation on (foo5.txt), and an obtained result is 7000, and belongs to a takeover range of the lock server 4. Therefore, the lock server 4 processes the reclaim lock request.

When a host needs to perform an operation on a file in the distributed system, the host sends a lock request to a protocol server by using an application on the host. The protocol server sends the lock request to a corresponding lock proxy. The lock proxy performs hash calculation on a file identifier (the identifier is, for example, an FSID or an FID) carried in the lock request, determines, according to a calculation result, a lock server whose management range includes the file, and sends the lock request to the lock server for corresponding processing. A hashing algorithm used to perform hash calculation on the file identifier needs to be the same as the hashing algorithm used to generate the consistent hashing ring. For example, a file identifier carried in the lock request is (foo2.txt), and the lock proxy performs hash calculation on the file identifier (foo2.txt). An obtained result is 6500, and it can be seen that, as shown in FIG. 2, the result falls within a range between the lock server 1 and the lock server 2 in the consistent hashing ring, the range is the management range of the lock server 2, and the lock request is processed by the lock server 2.

When a fault occurs in the lock server 2, the lock proxy identifies the lock server 2 in the consistent hashing ring as faulty. In this case, after receiving a lock request, the lock proxy performs hash calculation on file information (foo3.txt) carried in the lock request, and an obtained result is 7500 and falls within a range between the lock server 1 and the lock server 2 in the consistent hashing ring. However, the lock server 2 is in a faulty state. A takeover lock server of the lock server 2 is the lock server 4 according to the consistent hashing ring, that is, the range is the management range of the lock server 4, and therefore the lock proxy sends the lock request to the lock server 4 for processing.

An existing technology may be used in the method for obtaining the consistent hashing ring by using the consistent hashing algorithm according to names of nodes or IDs of lock servers. Details are not further described herein.

Based on the lock server management range and the lock server takeover relationship in the distributed system shown in FIG. 2, an embodiment of the present invention provides a method for processing lock permission in a distributed system. The method embodiment is applied to a lock server. A method implementation procedure of a protocol server and a lock proxy mentioned in this method embodiment is the same as the method described above, and is not separately described in this method embodiment again. A specific procedure is shown in FIG. 3.

The method may be applied to the distributed system shown in FIG. 1. In the distributed system in this embodiment of the present invention, there are four lock servers: a lock server 1, a lock server 2, a lock server 3, and a lock server 4. A quantity of the lock servers in this embodiment is merely an example, and a specific quantity is subject to an actual service requirement. An implementation principle thereof is the same as that in this embodiment.

Referring to FIG. 3, an example in which a takeover lock server is a first lock server and a lock server in which a fault occurs is a second lock server is used below to specifically describe processing of an acquire lock request.

Step 301: When a fault occurs in a lock server in the distributed system, a management node broadcasts a notification message to the lock servers in the distributed system. In this embodiment, the lock server in which the fault occurs is a second lock server. Therefore, the notification message carries an ID of the second lock server as identification information of the second lock server. To differ from another notification message, a notification message that a fault occurs in a lock server is referred to as a first notification message.

The lock server that receives the first notification message determines, according to the ID carried in the first notification message and a locally stored lock server management range, whether the lock server is a takeover lock server of the second lock server. If the lock server is the takeover lock server of the second lock server, the lock server enters a silent state for a resource for which the second lock server has already assigned permission; if the lock server is not the takeover lock server of the second lock server, the lock server does not enter a silent state.

In addition, after receiving the first notification message, the first lock server may enable a timer. When the timer expires after a preset time, the first lock server exits the silent state, and updates a takeover relationship of the first lock server.

Another fault detection manner is: The takeover lock server regularly sends a detection message to a corresponding lock server according to information about the takeover relationship, and after detecting that a fault occurs in the corresponding lock server, the takeover lock server enters a silent state.

Step 302: The first lock server receives an acquire lock request, where the acquire lock request carries an identifier of a target resource. The target resource is a resource that needs to be locked, and is a requested object of the acquire lock request, or in other words, is a resource for which lock permission waits to be assigned.

When performing an operation such as a read or write operation on a resource in a storage system, a host needs to send an acquire lock request to a lock server by using a protocol server and a lock proxy, where the acquire lock request carries a resource identifier of a resource that the host

requests to lock. The resource identifier may be an ID of a file that needs to be operated or an ID of a logical unit number (Logical Unit Number, LUN) that needs to be operated, and the acquire lock request requests permission to read a resource or to write to a resource.

The first lock server determines, according to the resource identifier, whether the requested resource belongs to a management range of the first lock server. For example, a hash calculation is performed on the resource identifier. If an obtained value belongs to a hash value range preset by the first lock server, the requested resource belongs to the management range of the first lock server; or if an obtained value does not belong to a hash value range preset by the first lock server, the requested resource does not belong to the management range of the first lock server.

The target resource that the acquire lock request requests to lock is located in the storage system, and the lock server manages lock permission for the target resource. This process is already described above, and is not separately described herein again.

There are many methods for sending, to the first lock server for processing, a lock request that is originally sent to the second lock server. For example, a router is directly disposed for the lock server and the host, and the router records the takeover relationship. When a fault occurs in the second lock server, the router sends, to the takeover lock server of the second lock server, the lock request that is originally sent to the second lock server. If no fault occurs in a lock proxy in a same node together with the second lock server, the foregoing solution may be used, and the lock proxy sends, to the takeover lock server of the second lock server, the lock request that is originally sent to the second lock server.

In the distributed system shown in FIG. 1, it is assumed that a fault occurs in the lock server 1, that is, the lock server 1 is the second lock server. According to the lock server management range and the lock server takeover relationship shown in FIG. 2, the lock server 2 is the takeover lock server of the lock server 1, and then the first lock server herein is the lock server 2.

Step 303: The first lock server queries a first resource information record list, where the first resource information record list records a resource identifier of a resource for which the second lock server has assigned lock permission.

The takeover lock server of the second lock server is in a silent state. After receiving an acquire lock request, each lock server including the first lock server first determines whether the lock server is currently in a silent state. If the lock server is in a silent state, each lock server further performs determining according to an identifier carried in the acquire lock request. If it is learned that the acquire lock request is an acquire lock request that belongs to a takeover range of the lock server, step 303 is performed. If it is not the first lock server but another lock server that receives the acquire lock request, skip step 303 and exit the entire procedure.

to the silent range, query the first resource information record list; and if the first resource information record list does not record the identifier of the first resource, assign lock permission for the first resource according to the first acquire lock request.

Optionally, after the lock request management apparatus 4 enters the silent state, the receiving module 41 is further configured to receive a second acquire lock request, where the second acquire lock request is used to request to lock a second resource, and the second acquire lock request carries an identifier of the second resource; and the lock request processing module 43 is further configured to: after detecting that the second resource belongs to a management range of the lock request management apparatus 4, assign lock permission for the second resource according to the second acquire lock request.

Optionally, after the lock request management apparatus 4 enters the silent state, the receiving module 41 is further configured to receive a third acquire lock request, where the third acquire lock request is used to request to lock a third resource, and the third acquire lock request carries an identifier of the third resource; the lock request processing module 43 is further configured to: after detecting that the third resource belongs to the management range of the another lock request management apparatus, query the first resource information record list; and if the first resource information record list has recorded the resource identifier of the resource requested in the third acquire lock request, refuse to assign lock permission for the third resource according to the third acquire lock request.

Optionally, after the lock request management apparatus 4 enters the silent state, the lock request processing module 43 is further configured to: receive a reclaim lock request, where the reclaim lock request carries an identifier of a fourth resource and permission that is assigned by the another lock request management apparatus for the fourth resource, and the fourth resource is a resource for which the another lock request management apparatus has assigned permission; and reassign the same permission for the fourth resource according to the permission that has been assigned by the another lock request management apparatus.

Optionally, the storage module 42 is further configured to receive a first notification message, where the first notification message carries identification information of the another lock request management apparatus; the receiving module 41 is further configured to: after it is determined, according to an identifier of the another lock request management apparatus and a takeover relationship of the lock request management apparatus 4, that the lock request management apparatus is a takeover lock request management apparatus 4 of the another lock request management apparatus, send the lock management range of the another lock request management apparatus to the storage module 42; and that the foregoing storage module 42 is configured to store the lock

management range of the another lock request management apparatus specifically includes: the storage module 42 is configured to receive the lock management range of the another lock request management apparatus from the receiving module 41, and store the lock management range.

Optionally, the lock request management apparatus 4 may further include a protocol server module 45 and a lock proxy module 46. The protocol server module 45 is configured to: receive a packet from a host, parse out the first acquire lock request from the packet, and forward the first acquire lock request to the lock proxy module 46. The lock proxy module 46 is configured to: perform determining according to the identifier of the first resource carried in the first acquire lock request; and when determining that the lock request processing module 43 manages the first resource, send the first acquire lock request to the lock request processing module 43 by using the receiving module 41.

Optionally, the silence module 44 is further configured to: after permission is reassigned for all resources for which the another lock request management apparatus has assigned permission, enable the lock request management apparatus to exit the silent state; or after a preset time, enable the lock request management apparatus 4 to exit the silent state.

Optionally, the storage module 42 is further configured to: after the lock request management apparatus 4 exits the silent state, update the management range of the lock request management apparatus, where an updated management range of the lock request management apparatus includes the management range of the lock request management apparatus and the management range of the another lock request management apparatus.

An embodiment of the present invention further provides another server 5 in a distributed system, and a structure of the server 5 is shown in FIG. 5. The server 5 includes: an interface 51, a memory 52, and a processor 53. The server 5 may execute the method in the method embodiment. Specifically, the processor 53 of the server 5 performs steps of the method. The interface 51 provides an external data interface, and the memory 52 provides data storage space. The server 5 is only briefly described below. For specific content, refer to the foregoing description.

The interface 51 provides an external interface to, for example, receive an acquire lock request or a reclaim lock request.

The memory 52 is configured to store a first resource information record list, where the first resource information record list records a resource identifier of a resource for which the another server has assigned lock permission. It may be learned from the method embodiment that the memory 52 may be further configured to store other information, for example, a second resource information record list or a detailed resource information record list; and the memory 52 may be further configured to store the lock management

range of the another server.

The processor 53 is configured to run a program to perform steps in the method embodiment: for example, after learning that a fault occurs in the another server, setting the server 5 to a silent state, where a silent range of the silent state is the resource for which the another server has assigned permission; receiving a first acquire lock request, where the first acquire lock request is used to request to lock a first resource, and the first acquire lock request carries an identifier of the first resource; detecting that the first resource belongs to the management range of the another lock server; querying, by the first lock server, the first resource information record list; and if the first resource information record list does not record the identifier of the first resource, assigning, by the first lock server, lock permission for the first resource according to the first acquire lock request.

Operations in the method embodiment, for example, entering the silent state, exiting the silent state, querying, determining, and assigning permission, may be all executed by the processor 53.

Optionally, the server 5 may further include a protocol server module 54 and a lock proxy module 55.

The protocol server module 54 is configured to: receive a packet from a host, and parse out the first acquire lock request from the packet; and is further configured to forward the first acquire lock request to the lock proxy module 55. The lock proxy module 55 is configured to: perform determining according to the identifier of the first resource carried in the first acquire lock request; and when determining that the server 5 manages the first resource, send the first acquire lock request to the interface 51.

A form of a "list" mentioned in each embodiment of the present invention, for example, the first resource information record list, is not limited to a sheet or a list, but content stored in the "list" is used as a definition of the "list".

Each aspect of the present invention or a possible implementation manner of each aspect may be specifically implemented as a system, a method, or a computer program product. Therefore, each aspect of the present invention or a possible implementation manner of each aspect may use a form of a complete hardware embodiment, a complete software embodiment (including firmware, resident software, and the like), or an embodiment combining software and hardware, which are collectively referred to as a "circuit", a "module", or a "system" herein. In addition, each aspect of the present invention or the possible implementation manner of each aspect may use a form of a computer program product, where the computer program product refers to computer readable program code stored in a computer readable medium.

The computer readable medium may be a computer readable signal medium or a computer readable storage medium. The computer readable storage medium includes but is not limited to electronic,