METHOD FOR IMPLEMENTING TIMER PROCESSING IN RDMA, SYSTEM, CHIP, AND STORAGE MEDIUM

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is a Regional Phase of International Patent Application No. PCT/CN2024/104676, filed on July 10, 2024, which claims priority to Chinese Patent Application No. 202410355510.4, entitled "METHOD AND SYSTEM FOR IMPLEMENTING TIMER PROCESSING IN RDMA, CHIP, AND STORAGE MEDIUM" and filed with the China National Intellectual Property Administration on March 26, 2024, the entire contents of which are incorporated herein by reference.

TECHNICAL FIELD

[0002] The present application relates to the field of data storage technologies, and in particular, to a method and a system for implementing timer processing in remote direct memory access (RDMA), a chip, and a storage medium.

BACKGROUND

[0003] In the application of RDMA technology, each queue pair (QP) within a respective function may be provided with time nodes, which are used for timing and waking up the RDMA device to perform corresponding transmission and reception processing. In some examples, the RDMA engine may provide time nodes for each QP, the time nodes involving two types of timers: namely an acknowledge (Ack) timer and a Receiver not ready (Rnr) timer.

[0004] Generally, timing functions of the above time nodes are implemented using time wheels. Taking the Ack timer for instance, in an application scenario, the RDMA engine provides multiple (e.g., 4) time wheels. The time wheels may represent durations of 2ms, 32ms, 512ms, and 8s, respectively. Each time wheel includes 16 scales, so correspondingly, a duration represented by each scale on a respective time wheel is 128μs, 2ms, 32ms, and 512ms. Since the RDMA engine supports a large number of QPs, typically millions of QPs, the RDMA engine is required to provide up to 1 million time nodes of the ACK type. The RDMA engine manages the time nodes by means of linked lists. The RDMA engine provides one linked list for each time wheel scale. When a particular time wheel scale expires, the RDMA engine may read all time nodes in the linked list corresponding to the scale, for the RDMA engine to perform timeout determination. In extreme cases, 1 million time nodes may be generated and linked into the linked list corresponding to each scale on each time wheel. In consideration of the power performance area (PPA), usually it is not possible to provide a large amount of resources on-chip to store the 1 million linked list nodes. Linked list nodes that cannot be stored on the chip may overflow into the double data rate (DDR)

memory of the host. That is, the RDMA engine provides one linked list having on-chip and off-chip parts to manage all time nodes corresponding to one time wheel scale.

[0005] In the related art, there are two methods. In the first method, four time wheels are shared, which has the following disadvantages. When the RDMA engine is required to support virtualization, since the nodes in the linked list corresponding to one time wheel scale may belong to different functions, it is impossible to provide only one linked list on-chip to store time nodes of different functions. If there is no sufficient idle space inside the chip, time nodes need to be linked into the linked list in the host, which may easily cause the problem of function interference. In the other method, a separate set of time wheels are provided for each function, so that each time wheel scale of each function has an independent linked list to manage respective time nodes. That is, time nodes of different functions may not be linked into the same linked list.

[0006] FIG. 1 illustrates a schematic diagram of a principle supporting virtualization management of time nodes of multiple functions provided in the related art. As can be seen, a set of time wheels (4 time wheels in total) manage 64 linked lists, which occupies a large PPA. When the RDMA engine supports virtualization, the RDMA engine provides a set of time wheels for each function. A larger number of functions are supported during virtualization, generally thousands of functions. Thus, the RDMA engine implements a total of n×4 time wheels, implements and manages n×4×64 on-chip linked lists, and is simultaneously responsible for managing off-chip parts of these linked lists. Here, n represents the number of the functions. Therefore, the RDMA engine consumes a large quantity of resources for timers, which may affect the PPA.

[0007] Additionally, since the linked list used by the time wheel is split into an on-chip part and an off-chip part, the overall design becomes complicated. Furthermore, the time wheel implemented by the RDMA engine can provide only a maximum timeout of 8s. When the RDMA engine requires a timeout duration exceeding 8s, for example, a long-distance RDMA engine is required to support longer timeout time, the RDMA engine is required to add more time wheels to achieve this purpose, which increases the PPA and leads the overall architecture to less flexibility.

SUMMARY

[0008] According to various embodiments of the present application, a method and a system for implementing timer processing in RDMA, a chip, and a storage medium are provided.

[0009] In one aspect of the present application, a method for implementing timer processing in RDMA is provided, applied to an RDMA engine, including at least the following steps:

receiving an RDMA message generated by a host, generating a corresponding timeout-time node for the message, and in a case where a timeout time of the timeout-time node does not exceed a preset threshold, determining a corresponding time wheel in a shared time wheel group for the

timeout-time node, and determining a timeout-time scale corresponding to the timeout time in the determined time wheel for the timeout-time node;

determining whether an idle node exists in a storage unit corresponding to the shared time wheel group;

applying for one idle node in a case where the idle node exists in the storage unit, to store node content of the timeout-time node, and linking the idle node into a linked list corresponding to the determined timeout-time scale, to enable the RDMA engine to perform timeout determination; and

notifying the host of the timeout-time node by means of an event queue in a case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit, to enable the host to manage the timeout time.

[0010] The method further includes: configuring, in the RDMA engine, a shared time wheel group. The time wheel group includes a first predetermined number of different categories of time wheels, each time wheel corresponds to a second predetermined number of scale values, and each scale value is associated with a linked list.

[0011] The applying for one idle node in the case where the idle node exists in the storage unit, to store the node content of the timeout-time node, and linking the idle node into the linked list corresponding to the determined timeout-time scale further includes:

performing idle node applying in a case where the idle node exists in the storage unit, to obtain an idle storage address;

writing a function ID and a queue pair number (QPN) of the RDMA message into a storage space corresponding to the idle storage address; and

establishing, in the linked list corresponding to the determined timeout-time scale, an association between the determined timeout-time scale and the idle storage address.

[0012] The RDMA engine performing timeout determination further includes:

traversing a corresponding linked list when a time wheel scale expires and retrieving all timeout-time nodes of the linked list; and

sending the retrieved timeout-time nodes to the RDMA engine for timeout determination, and executing a task in a corresponding timeout-time node.

[0013] The notifying the host of the timeout-time node by means of the event queue in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit, to enable the host to manage the timeout time further includes:

notifying the host of an ID of a function and QPN information of an RDMA corresponding to the timeout-time node by means of the event queue corresponding to the function in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit.

[0014] Correspondingly, in another aspect of the present application, a system for implementing timer processing in RDMA is further provided, including at least:

an RDMA message receiving and processing unit, configured to receive an RDMA message generated by a host, generate a corresponding timeout-time node for the message, and in a case where a timeout time of the timeout-time node does not exceed a preset threshold, determine, for the timeout-time node, a corresponding time wheel in a shared time wheel group and a timeout-time scale corresponding to the timeout time in the time wheel;

a condition determination unit, configured to determine whether an idle node exists in a storage unit corresponding to the shared time wheel group;

a first processing unit, configured to apply for one idle node in a case where the idle node exists in the storage unit, to store node content of the timeout-time node, and link the idle node into a linked list corresponding to the determined timeout-time scale, to enable an RDMA engine to perform timeout determination; and

a second processing unit, configured to notify the host of the timeout-time node by means of an event queue in a case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit.

[0015] The system further includes:

a configuration unit, configured to configure, in the RDMA engine, a shared time wheel group, where the time wheel group comprises a first predetermined number of different categories of time wheels, each time wheel corresponds to a second predetermined number of scale values, and each scale value is associated with a linked list.

[0016] The first processing unit further includes:

an idle node application unit, configured to apply for, in the case where the idle node exists in the storage unit, one idle node to obtain an idle storage address;

a writing unit, configured to write a function ID and a queue pair number, QPN, of the RDMA message into a storage space corresponding to the idle storage address;

a linking unit, configured to establish, in the linked list corresponding to the determined timeout-time scale, an association between the determined timeout-time scale and the idle storage address;

a traversal processing unit, configured to traverse a corresponding linked list when a time wheel scale expires and retrieve all timeout-time nodes of the linked list; and

a first timeout processing unit, configured to send the retrieved timeout-time nodes to the RDMA engine for timeout determination, and execute a task in a corresponding timeout-time node.

[0017] The second processing unit further includes:

a notification unit, configured to notify the host of an ID a function and QPN information of an RDMA message corresponding to the timeout-time node by means of the event queue corresponding to the function in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit.

[0018] Correspondingly, in another aspect of the present application, a chip is further provided.

The chip is deployed with the system as described above.

[0019] Correspondingly, in yet another aspect of the present application, a computer-readable storage medium is further provided, having a computer program stored thereon. The computer program, when executed by a processor, implements steps of the method as described above.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] To more clearly illustrate the technical solutions in the embodiments of the present application or the prior art, the following briefly introduces the drawings required for describing the embodiments or the prior art. Apparently, the drawings in the following description are only some embodiments of the present application. For those of ordinary skill in the art, other drawings obtained according to these drawings without creative efforts still fall within the scope of the present application.

[0021] FIG. 1 is a schematic diagram of a principle supporting virtualization management of time nodes of a plurality of functions provided in the prior art;

[0022] FIG. 2 is a main schematic flowchart of an embodiment of a method for implementing timer processing in RDMA according to the present application;

[0023] FIG. 3 is a schematic diagram of an application environment involved in the present application;

[0024] FIG. 4 is a schematic diagram of a principle of virtualization management of time nodes in the method according to the present application;

[0025] FIG. 5 is a schematic structural diagram of an embodiment of a system for implementing timer processing in RDMA according to the present application;

[0026] FIG. 6 is a schematic structural diagram of a first processing unit in FIG. 5; and

[0027] FIG. 7 is a schematic structural diagram of a second processing unit in FIG. 5.

DETAILED DESCRIPTION

[0028] To make objectives, technical solutions, and advantages of the present application clearer, the present application is detailed hereinafter with reference to accompanying drawings.

[0029] FIG. 2 illustrates a main schematic flowchart of a method for implementing timer processing in RDMA according to an embodiment of the present application. Referring to FIG. 2 to FIG. 4 together, in the embodiment, the method is applied to an RDMA communication system. Generally, as shown in FIG. 3, an RDMA communication system at least includes: a local host, a local external memory (DDR), a local RDMA engine, a local Peripheral Component Interconnect Express (PCIe) device, a remote PCIe device, a remote RDMA engine, a remote host, and a remote external storage space (DDR). The local PCIe device and the remote PCIe device may be, for example, network adapters, storage controllers, accelerator cards, hardware encryption cards, or

the like.

[0030] Specifically as shown in FIG. 2, in the embodiment of the present application, the method for implementing timer processing in RDMA is applied to the local RDMA. Specifically, the method further includes the following steps.

[0031] In step S10, an RDMA message generated by a host is received, a corresponding timeout-time node is generated for the message, and it is determined whether a timeout time of the timeout-time node exceeds a preset threshold; in a case where the timeout time does not exceed the preset threshold, a corresponding time wheel in a shared time wheel group is determined for the timeout-time node, and a timeout-time scale corresponding to the timeout time in the determined time wheel is determined for the timeout-time node.

[0032] It may be understood that in the embodiment of the present application, the following step is required to be preliminarily included:

[0033] configuring, in the RDMA engine, a shared time wheel group, the time wheel group including a first predetermined number of different categories of time wheels, each time wheel corresponding to a second predetermined number of scale values, each scale value corresponding to a linked list. As shown in FIG. 4, in the embodiment of the present application, the RDMA engine implements only one time wheel group and internally provides a number of time nodes comparable to the number of typical online flows. Specifically, in one example, the first number is 4, and the second number is 16. Therefore, in the example of FIG. 4, the RDMA engine implements a total of 4 time wheels, realizing and managing 4×64 on-chip linked lists, and the linked lists do not require any off-chip parts. In a specific example, a storage unit is utilized to store and manage the linked lists of the shared time wheel group.

[0034] In one example, it is assumed that the timeout time is 2s and a timeout time threshold is set to 8s. Since 2s is shorter than 8s, the threshold is not exceeded. Then, according to the timeout time of 2s, a corresponding time wheel is selected. It is assumed that the four time wheels in the corresponding time wheel group are: 16 s (Time Wheel 0), 16×16 s (Time Wheel 1), 256×16 s (Time Wheel 2), and 256×16×16 s (Time Wheel 3)); then, Time Wheel 0 is selected. If a current scale of Time Wheel 0 is at Scale 5, the timeout-time scale is calculated as: 5+2=7. That is, Scale 7 of Time Wheel 0 is the timeout-time scale.

[0035] In step S11, determination is performed on the timeout time and all nodes in a storage unit corresponding to a current shared time wheel group, and it is determined whether an idle node exists in the storage unit corresponding to the shared time wheel group.

[0036] In step S12, if a determination result is that the idle node exists in the storage unit, one idle node is applied for to store node content of the timeout-time node and is linked into a linked list corresponding to the determined timeout-time scale, to enable the RDMA engine to perform timeout determination.

[0037] In a specific example, step S12 further includes:

[0038] performing idle node applying in a case where the idle node exists in the storage unit, to obtain an idle storage address;

[0039] writing a function ID and a QPN of the RDMA message into a storage space corresponding to the idle storage address;

[0040] establishing, in the linked list corresponding to the determined timeout-time scale, an association between the determined timeout-time scale and the idle storage address; where in a specific example, a linked list or a bitmap may be utilized to manage idle nodes, each time scale corresponds to one linked list, and each linked list is managed through head and tail pointers of the linked list;

[0041] traversing a corresponding linked list when a time wheel scale expires and retrieving all timeout-time nodes of the linked list; and

[0042] sending the retrieved timeout-time nodes to the RDMA engine for timeout determination, and executing a task in a corresponding timeout-time node.

[0043] In step S13, the host is notified of the timeout-time node by means of an event queue in a case where the determination result is that the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit, to enable the host to manage the timeout time.

[0044] In a specific example, step S13 further includes:

[0045] notifying the host of a function ID and QPN information of an RDMA message corresponding to the timeout-time node by means of the event queue corresponding to the function in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit.

[0046] In a specific example, the host side may use a system timer thereof to provide a timing function for each received time node.

[0047] When the system timer expires, the host notifies the RDMA engine via a doorbell, to perform timeout determination on a corresponding time node and execute a task in the corresponding time node.

[0048] Referring to FIG. 4, a specific example is used for illustration. In the embodiment, the RDMA engine implements only one time wheel group and internally provides a number of time nodes comparable to the number of typical online flows. When time node resources are exhausted or when a timeout value of the timer exceeds a configured time value, the RDMA engine notifies a host-side driver by means of an event queue of each function, and the driver uses the timer provided by the system to implement the timing function. When the system timer expires, the host-side driver notifies the RDMA engine through a doorbell, to enable the RDMA engine to perform timeout determination. The RDMA engine implements a total of 4 time wheels, realizing and managing 4×64 on-chip linked lists. The linked lists do not require any off-chip parts.

[0049] In the solution, a process of using the timer by the RDMA engine is described as follows.

[0050] In step 1, the RDMA engine sends out a message and is required to monitor whether the message is completed within a certain period of time. Therefore, a timer is started in a timer management module of the RDMA engine. The timer management module determines a time wheel and a time wheel scale within time wheel according to a timeout value of the timer.

[0051] In step 2, the timer management module allocates time nodes from shared time node resources.

[0052] In step 3, the timer management module links the time nodes into a corresponding time wheel scale linked list.

[0053] In step 4, when the time wheel scale expires, the timer management module may traverse the corresponding linked list, retrieve all time nodes, and send the time nodes to the RDMA engine for timeout determination. For a time node that has timed out, a task carried by the time node is executed.

[0054] In step 5, when the shared time node resources are insufficient or when the timeout value of the timer exceeds a preset threshold, a host side (e.g., a host driver) is notified of the time node by means of an event queue of a function corresponding to the time node.

[0055] In step 6, the host driver uses the timer provided by the system to provide a timing function for the RDMA engine.

[0056] In step 7, when the system timer expires, the host driver notifies the RDMA engine via a doorbell to perform timeout determination.

[0057] It may be understood that, by implementing the method provided by the present application, an RDMA engine virtualization timer is achieved through a combination of software and hardware. Only the time nodes of typical online flows are retained on the chip. In most scenarios, the demand for time nodes is low and thus it is not necessary for the driver to interfere. When the on-chip time nodes are insufficient or the timeout value of the timer is large, the host is notified by means of an event queue. The host driver uses the timer provided by the system to implement the timing function.

[0058] In the technical solution, the RDMA engine may be allowed to support virtualization without requiring additional resources to implement the timer, thereby saving the PPA of the chip and simplifying the design of the chip. Furthermore, since a timer with a relatively large timeout value is directly sent to the host driver to be implemented by the system timer, the RDMA engine can support a wider range of timeout values.

[0059] FIG. 5 is a schematic structural diagram of a system for implementing timer processing in RDMA according to an embodiment of the present application, which is implemented in the application environment in FIG. 3. Referring to FIG. 6 to FIG. 7 together, in the embodiment, the system 1 includes at least:

[0060] an RDMA message receiving and processing unit 10, configured to receive an RDMA message generated by a host, generate a corresponding timeout-time node for the message, and in a case where a timeout time of the timeout-time node does not exceed a preset threshold, determine, for the timeout-time node, a corresponding time wheel in a shared time wheel group and a timeout-time scale corresponding to the timeout time in the time wheel;

[0061] a condition determination unit 11, configured to determine whether an idle node exists in a storage unit corresponding to the shared time wheel group;

[0062] a first processing unit 12, configured to apply for one idle node in a case where the idle node exists in the storage unit, to store node content of the timeout-time node, and link the idle node into a linked list corresponding to the determined timeout-time scale, to enable an RDMA engine to perform timeout determination;

[0063] a second processing unit 13, configured to notify the host of the timeout-time node by means of an event queue in a case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit, to enable the host to manage the timeout time; and

[0064] a configuration unit 14, configured to configure, in the RDMA engine, a shared time wheel group, the time wheel group including a first predetermined number of different categories of time wheels, each time wheel corresponding to a second predetermined number of scale values, each scale value being associated with a linked list.

[0065] As shown in FIG. 6, in a specific example, the first processing unit 12 further includes:

[0066] an idle node application unit 120, configured to apply for, in the case where the idle node exists in the storage unit, one idle node to obtain an idle storage address;

[0067] a writing unit 121, configured to write a function identifier, ID, and a queue pair number, QPN, of the RDMA message into a storage space corresponding to the idle storage address;

[0068] a linking unit 122, configured to establish, in the linked list corresponding to the determined timeout-time scale, an association between the determined timeout-time scale and the idle storage address;

[0069] a traversal processing unit 123, configured to traverse a corresponding linked list when a time wheel scale expires and retrieve all timeout-time nodes of the linked list; and

[0070] a first timeout processing unit 124, configured to send the retrieved timeout-time nodes to the RDMA engine for timeout determination, and execute a task in a corresponding timeout-time node.

[0071] As shown in FIG. 7, in a specific example, the second processing unit 13 further includes:

[0072] a notification unit 130, configured to notify the host of a function ID and QPN information of an RDMA message corresponding to the timeout-time node by means of the event queue corresponding to the function in the case where the timeout time exceeds the preset threshold

or no idle nodes exist in the storage unit.

[0073] Correspondingly, in another aspect of the present application, a chip is further provided, the chip being deployed with the system as described above in FIG. 5 to FIG. 7. For more details, please refer to and combine the foregoing description of FIG. 5 to FIG. 7, which will not be repeated herein.

[0074] Correspondingly, in yet another aspect of the present application, a computer-readable storage medium is further provided, having a computer program stored thereon. The computer program, when executed by a processor, implements steps of the method as described above in FIG. 2 to FIG. 4. For more details, please refer to and combine the foregoing description of FIG. 2 to FIG. 4, which will not be repeated herein.

[0075] Implementing the embodiments of the present application can have the following beneficial effects.

[0076] The present application provides a method and a system for implementing timer processing in RDMA, a chip, and a storage medium. An RDMA engine virtualization timer is achieved through a combination of software and hardware. Only the time nodes of typical online flows are retained on the chip. In most scenarios, the demand for time nodes is low, and thus it is not necessary for the driver to interfere. When the on-chip time nodes are insufficient or the timeout value of the timer is large, the host is notified by means of an event queue. The host driver uses the timer provided by the system to implement the timing function.

[0077] In the technical solution, the RDMA engine may be allowed to support virtualization without requiring additional resources to implement the timer, thereby saving the PPA of the chip and simplifying the design of the chip. Furthermore, since a timer with a relatively large timeout value is directly sent to the host driver to be implemented by the system timer, the RDMA engine can support a wider range of timeout values, thereby improving architectural flexibility.

[0078] Those skilled in the art should understand that the embodiments of the present application may be provided as methods, apparatuses, or computer program products. Therefore, the present application may be embodied in the form of hardware-only, software-only, or combination of software and hardware aspects. Furthermore, the present application may be embodied in the form of a computer program product implemented on one or more computer-usable storage media (including, but not limited to, a disk storage, a CD-ROM, an optical storage, and the like) having computer-usable program codes thereon.

[0079] The present application is described with reference to the flowcharts and/or block diagrams of the methods, apparatus (systems), and computer program products according to the embodiments of the present application. It should be understood that each process and/or block in the flowcharts and/or block diagrams, as well as combinations of processes and/or blocks in the flowcharts and/or block diagrams, can be implemented by computer program instructions. These

computer program instructions may be provided to a processor of a general-purpose computer, a special-purpose computer, an embedded processor, or another programmable data processing apparatus to produce a machine, such that instructions executed by the processor of the computer or another programmable data processing apparatus generate an apparatus configured to implement the functions specified in one or more processes of the flowchart and/or one or more blocks of the block diagram.

[0080] The above disclosure is merely a preferred embodiment of the present application, and certainly cannot be used to limit the scope of the present application.

CLAIMS

1. A method for implementing timer processing in remote direct memory access, RDMA, applied to an RDMA engine, comprising:

receiving an RDMA message generated by a host, generating a corresponding timeout-time node for the message, and in a case where a timeout time of the timeout-time node does not exceed a preset threshold, determining a corresponding time wheel in a shared time wheel group for the timeout-time node, and determining a timeout-time scale corresponding to the timeout time in the determined time wheel for the timeout-time node;

determining whether an idle node exists in a storage unit corresponding to the shared time wheel group;

applying for one idle node in a case where the idle node exists in the storage unit, to store node content of the timeout-time node, and linking the idle node into a linked list corresponding to the determined timeout-time scale, to enable the RDMA engine to perform timeout determination; and

notifying the host of the timeout-time node by means of an event queue in a case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit, to enable the host to manage the timeout time.

2. The method according to claim 1, further comprising:

configuring, in the RDMA engine, a shared time wheel group, wherein the time wheel group comprises a first predetermined number of different categories of time wheels, each time wheel corresponds to a second predetermined number of scale values, and each scale value is associated with a linked list.

3. The method according to claim 2, wherein the applying for one idle node in the case where the idle node exists in the storage unit, to store the node content of the timeout-time node, and linking the idle node into the linked list corresponding to the determined timeout-time scale further comprises:

performing idle node applying in a case where the idle node exists in the storage unit, to obtain an idle storage address;

writing a function identifier, ID, and a queue pair number, QPN, of the RDMA message into a storage space corresponding to the idle storage address; and

establishing, in the linked list corresponding to the determined timeout-time scale, an association between the determined timeout-time scale and the idle storage address.

4. The method according to claim 3, wherein each time scale corresponds to one linked list, and

a head pointer and a tail pointer of each linked list are configured to manage the linked list.

5. The method according to claim 3 or 4, wherein the RDMA engine performing timeout determination further comprises:

traversing a corresponding linked list when a time wheel scale expires and retrieving all timeout-time nodes of the linked list; and

sending the retrieved timeout-time nodes to the RDMA engine for timeout determination, and executing a task in a corresponding timeout-time node.

6. The method according to claim 5, wherein notifying the host of the timeout-time node by means of the event queue in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit, to enable the host to manage the timeout time further comprises:

notifying the host of an ID of a function and QPN information of an RDMA message corresponding to the timeout-time node by means of the event queue corresponding to the function in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit.

7. The method according to claim 6, wherein the timeout time is determined by a system timer.

8. A system for implementing timer processing in remote direct memory access, RDMA, comprising:

an RDMA message receiving and processing unit, configured to receive an RDMA message generated by a host, generate a corresponding timeout-time node for the message, and in a case where a timeout time of the timeout-time node does not exceed a preset threshold, determine, for the timeout-time node, a corresponding time wheel in a shared time wheel group and a timeout-time scale corresponding to the timeout time in the time wheel;

a condition determination unit, configured to determine whether an idle node exists in a storage unit corresponding to the shared time wheel group;

a first processing unit, configured to apply for one idle node in a case where the idle node exists in the storage unit, to store node content of the timeout-time node, and link the idle node into a linked list corresponding to the determined timeout-time scale, to enable an RDMA engine to perform timeout determination; and

a second processing unit, configured to notify the host of the timeout-time node by means of an event queue in a case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit.

9. The system according to claim 8, further comprising:

a configuration unit, configured to configure, in the RDMA engine, a shared time wheel group, wherein the time wheel group comprises a first predetermined number of different categories of time wheels, each time wheel corresponds to a second predetermined number of scale values, and each scale value is associated with a linked list.

10. The system according to claim 9, wherein the first processing unit further comprises:

an idle node application unit, configured to apply for, in the case where the idle node exists in the storage unit, one idle node to obtain an idle storage address;

a writing unit, configured to write a function identifier, ID, and a queue pair number, QPN, of the RDMA message into a storage space corresponding to the idle storage address; and

a linking unit, configured to establish, in the linked list corresponding to the determined timeout-time scale, an association between the determined timeout-time scale and the idle storage address.

11. The system according to claim 10, wherein each time scale corresponds to one linked list, and a head pointer and a tail pointer of each linked list are configured to manage the linked list.

12. The system according to claim 10 or 11, wherein the first processing unit further comprises:

a traversal processing unit, configured to traverse a corresponding linked list when a time wheel scale expires and retrieve all timeout-time nodes of the linked list; and

a first timeout processing unit, configured to send the retrieved timeout-time nodes to the RDMA engine for timeout determination, and execute a task in a corresponding timeout-time node.

13. The system according to claim 8, wherein the second processing unit further comprises:

a notification unit, configured to notify the host of an ID of a function and QPN information of an RDMA message corresponding to the timeout-time node by means of the event queue corresponding to the function in the case where the timeout time exceeds the preset threshold or no idle nodes exist in the storage unit;

wherein the timeout time is determined by a system timer.

14. A chip, comprising the system according to any one of claims 8 to 13.

15. A computer-readable storage medium, having a computer program stored thereon, wherein the computer program, when executed by a processor, implements steps of the method according to any one of claims 1 to 7.