

METHOD AND APPARATUS FOR UPGRADING APPLICATION

TECHNICAL FIELD

5 [0001] The present application relates to the field of computer technologies, and in particular, to a method and an apparatus for upgrading an application.

BACKGROUND

[0002] Various services provided by a system for users are actually supported by various corresponding applications.

10 [0003] Usually, a running application receives backflow data from other data sources (such as systems, applications, clients, etc.) all the time. A recommendation scenario is used as an example. User A reads an article, and then the behavior data (namely, reading the article) flows back to a system by using a log. After parsing the log, the system queries friend information of user A, and then flows back the behavior data and the friend information of user A to a recommendation application. As such,
15 the recommendation application can recommend information similar to "Your friend A has read the XX article" to friends of user A based on the backflow data. This kind of application requires high availability because the application needs to provide services for users in real time. However, an application inevitably needs to be upgraded.

20 [0004] In the existing technology, an application upgrade procedure is shown in FIG. 1. In the upgrade process, an operation needs to be performed after manual confirmation in each phase. The procedure includes the following steps:

[0005] A1. Manually stop an old application from receiving backflow data from a data source.

25 [0006] A2. Manually submit a command (namely, a kill command) for disabling the old application to an application management module after manually confirming that processing of unprocessed data in the old application is completed.

[0007] A3. Manually submit a command (namely, a start command) for starting a

new application to the application management module after manually determining that the application management module completely disables the old application.

[0008] A4. Wait for the application management module to start the new application.

5 **[0009]** As described above, because the application is down for upgrade, a service cannot be provided for a user in the entire upgrade process until the new application is started. In addition, because human participation is needed in each phase, a relatively long time is needed for upgrading the entire application. As shown in FIG. 1, a time consumed between A1 and A2 is greater than 4 minutes, a time consumed between A2
10 and A3 is greater than 3 minutes, a time consumed between A3 and A4 is greater than 3 minutes, and the entire upgrade process needs to take more than 10 minutes.

[0010] In conclusion, in the existing technology, there is a problem that an application upgrade takes a relatively long time.

SUMMARY

15 **[0011]** The present application provides a method and an apparatus for upgrading an application, to alleviate a problem in the existing technology that an application upgrade takes a relatively long time.

[0012] According to a method for upgrading an application provided in an implementation of the present application, the method is applied to an application
20 management module, and the method includes the following: obtaining an application name of a submitted application; triggering a predetermined disable command when the submitted application has the same name as an existing application, to invoke an end function of each node in the existing application; recycling a process occupied by each node when receiving execution completion information returned by the existing
25 application; and triggering a predetermined start command when recycling is completed, to start a process that corresponds to each node in the submitted application.

[0013] Optionally, the triggering a predetermined disable command when the submitted application has the same name as an existing application, to invoke an end
30 function of each node in the existing application includes the following: determining whether a submitted upgrade command for the existing application is received when the submitted application has the same name as the existing application; and

triggering the predetermined disable command when the submitted upgrade command for the existing application is received, to invoke the end function of each node in the existing application.

[0014] Optionally, the application includes a JStorm application.

5 **[0015]** The process includes a jvm process.

[0016] According to a method for disabling an application provided in an implementation of the present application, the method is applied to an application to be upgraded, and the method includes the following: after an application management module invokes an end function of a node, stopping a root node from receiving
10 backflow data, and determining the root node as a current node; processing data in the current node and sending the data to a downstream node; broadcasting an ending identifier to all downstream sub-nodes of the current node after completing processing of the data in the current node and receiving ending identifiers broadcast by all upstream nodes; and when the current node has a next node, determining the next
15 node as the current node, and returning to perform the step of processing data in the current node and sending the data to a downstream node; or when the current node is the last node, returning, by the root node, execution completion information to the application management module to wait for the application management module to recycle a process that corresponds to each node.

20 **[0017]** Optionally, the application includes a JStorm application.

[0018] The process includes a jvm process.

[0019] According to an apparatus for upgrading an application provided in an implementation of the present application, the apparatus is applied to an application management module, and the apparatus includes the following: an acquisition unit,
25 configured to obtain an application name of a submitted application; an invocation unit, configured to trigger a predetermined disable command when the submitted application has the same name as an existing application, to invoke an end function of each node in the existing application; a recycling unit, configured to recycle a process occupied by each node when execution completion information returned by the
30 existing application is received; and an initiation unit, configured to trigger a predetermined start command when recycling is completed, to start a process that corresponds to each node in the submitted application.

[0020] Optionally, the invocation unit includes the following: a determining subunit, configured to determine whether a submitted upgrade command for the

existing application is received when the submitted application has the same name as the existing application; and an invocation subunit, configured to trigger the predetermined disable command when the submitted upgrade command for the existing application is received, to invoke the end function of each node in the existing application.

[0021] Optionally, the application includes a JStorm application.

[0022] The process includes a jvm process.

[0023] According to an apparatus for disabling an application provided in an implementation of the present application, the apparatus is applied to an application to be upgraded, and the apparatus includes the following: a receiving stopping unit, configured to: after an application management module invokes an end function of a node, stop a root node from receiving backflow data, and determine the root node as a current node; a data processing unit, configured to process data in the current node and send the data to a downstream node; an identifier broadcasting unit, configured to broadcast an ending identifier to all downstream sub-nodes of the current node after processing of the data in the current node is completed and ending identifiers broadcast by all upstream nodes are received; a control unit, configured to: when the current node has a next node, determine the next node as the current node, and then input data of the next node to the data processing unit; and an information returning unit, configured to: when the current node is the last node, return, by the root node, execution completion information to the application management module to wait for the application management module to recycle a process that corresponds to each node.

[0024] Optionally, the application includes a JStorm application.

[0025] The process includes a jvm process.

[0026] In the implementations of the present application, the application management module (such as a topology manager (TM)) is reconstructed, so that the application management module can allow a user to submit the application that has the same name as the existing application. Therefore, the application management module can automatically upgrade the existing application that has the same name when detecting that the user submits the application that has the same name. As such, the problem in the existing technology that a relatively long time is consumed because human participation is needed in an application upgrade process can be avoided, thereby reducing a time consumed for an application upgrade. In the implementations

of the present application, an application ending procedure is further reconstructed, so that any node needs to broadcast an ending identifier to all downstream nodes when completing processing of data in the node and receiving ending identifiers broadcast by all upstream nodes. The root node can return the execution completion information to the application management module only after all the nodes complete data processing, to wait for the application management module to recycle the process that corresponds to each node. Therefore, it can be ensured, by using the ending identifier, that processing of data in each node of the application is completed, to ensure the integrity of data processing and avoid omission that occurs when completion of data processing is determined manually. In addition, because there is no human participation, the time consumed for the application upgrade can be reduced.

BRIEF DESCRIPTION OF DRAWINGS

- [0027] FIG. 1 is a flowchart illustrating a method for upgrading an application in the existing technology;
- 15 [0028] FIG. 2 is a flowchart illustrating a method for upgrading an application, according to an implementation of the present application;
- [0029] FIG. 3 is a schematic diagram illustrating prompt information, according to an implementation of the present application;
- [0030] FIG. 4 is a schematic diagram illustrating nodes in an application, according to an implementation of the present application;
- 20 [0031] FIG. 5 is a flowchart illustrating a method for disabling an application, according to an implementation of the present application;
- [0032] FIG. 6 is a structural diagram of hardware of a device where an apparatus for upgrading an application is located, according to the present application;
- 25 [0033] FIG. 7 is a schematic diagram illustrating modules of an apparatus for upgrading an application, according to an implementation of the present application;
- [0034] FIG. 8 is a structural diagram of hardware of a device where an apparatus for disabling an application is located, according to the present application; and
- 30 [0035] FIG. 9 is a schematic diagram illustrating modules of an apparatus for disabling an application, according to an implementation of the present application.

DESCRIPTION OF IMPLEMENTATIONS

[0036] Example implementations are described in detail here, and examples of the implementations are presented in the accompanying drawings. When the following description relates to the accompanying drawings, the same numbers in different
5 accompanying drawings represent the same or similar elements unless specified otherwise. Implementations described in the following example implementations do not represent all implementations consistent with the present application. Instead, they are only examples of apparatuses and methods consistent with some aspects of the present application that are described in detail in the appended claims.

10 **[0037]** The terms used in the present application are merely for illustrating specific implementations, and are not intended to limit the present application. The terms "a" and "the" of singular forms used in the present application and the appended claims are also intended to include plural forms, unless otherwise specified in the context clearly. It should be further understood that the term "and/or" used in the
15 present specification indicates and includes any or all possible combinations of one or more associated listed items.

[0038] It should be understood that although terms "first", "second", "third", etc. can be used in the present application to describe various types of information, the information is not limited to the terms. These terms are only used to differentiate
20 between information of a same type. For example, without departing from the scope of the present application, first information can also be referred to as second information, and similarly, the second information can be referred to as the first information. Depending on the context, for example, the word "if" used here can be explained as "while", "when", or "in response to determining".

25 **[0039]** The following illustrates a JStorm scenario that the implementations of the present application can be applied to.

[0040] Storm is an open-source distributed fault-tolerant real-time computing system, and the complexity of parallel batch data processing can be reduced by using Storm. Generally, Storm is used in a system for real-time processing.

30 **[0041]** JStorm is a quasi-Storm system re-implemented by using JAVA. JStorm inherits all advantages of Storm, and JStorm is more powerful, more stable, and has better performance than Storm.

[0042] Usually, a JStorm application can be deployed in an application

management module (such as a topology manager (TM)) of a system. The JStorm application can communicate with the application management module, and complete functions such as an application upgrade.

[0043] FIG. 2 is a flowchart illustrating a method for upgrading an application, according to an implementation of the present application. The present implementation is described from the perspective of an application management module side, and includes the following steps.

[0044] Step 110: Obtain an application name of a submitted application.

[0045] In the present implementation, an application management module is reconstructed to allow a user to submit an application that has the same name as an existing application in the application management module.

[0046] After the user submits the application, the application management module can obtain the application name of the submitted application.

[0047] Step 120: Trigger a predetermined disable command when the submitted application has the same name as an existing application, to invoke an end function of each node in the existing application.

[0048] In the present implementation, if the application provided by the user has the same name as the existing application, it indicates that the existing application that has the same name needs to be upgraded, and the application manager automatically enters an upgrade procedure, that is, triggers the predetermined disable command to invoke the end function of each node in the existing application.

[0049] The end function is used by the node to process unprocessed data and wait to recycle a process that corresponds to the node.

[0050] Step 130: Recycle a process occupied by each node when receiving execution completion information returned by the existing application.

[0051] In the present implementation, after invoking the end function of each node in the existing application, the application management module needs to wait for each node in the existing application to process unprocessed data.

[0052] Usually, after completing processing of data, the existing application returns one piece of execution completion information. Therefore, when the application management module receives the execution completion information returned by the existing application, it indicates that all unprocessed data in the existing application has been processed. Therefore, the process occupied by each node can be recycled.

[0053] Step 140: Trigger a predetermined start command when recycling is completed, to start a process that corresponds to each node in the submitted application.

[0054] In the present implementation, after the application management module
5 recycles processes occupied by all nodes in the existing application, it indicates that the existing application is completely disabled, and then the submitted application can be started.

[0055] The predetermined start command is triggered when recycling is completed, to start the process that corresponds to each node in the submitted
10 application.

[0056] After the process that corresponds to each node in the submitted application is started, it indicates that the submitted application is started. As such, an entire application upgrade process ends.

[0057] According to the present implementation, the application management
15 module is reconstructed, so that the application management module can allow the user to submit the application that has the same name. Therefore, the application management module can automatically upgrade the existing application that has the same name when detecting that the user submits the application that has the same name. As such, a time consumed for an application upgrade can be reduced (which
20 can be shortened to about 3 minutes), and a problem in the existing technology that a relatively long time is consumed because human participation is needed in an application upgrade process can be avoided.

[0058] In the previous implementation, the application management module automatically upgrades the existing application that has the same name as long as the
25 application management module detects that the submitted application has the same name, that is, disables an old application (the existing application), and starts a new application (the submitted application). As such, although an application upgrade time can be shortened, application submission is operated manually in actual applications. The manual operation usually has many uncertainties. For example, although the
30 application submitted by the user has the same name, the application is not used to upgrade the old application, but is only another application that has the same name. In this case, the old application is directly upgraded by using the method for upgrading an application provided in the previous implementation. Consequently, an upgrade error occurs.

[0059] To alleviate such a problem, in another implementation of the present application, step 120 can include the following steps: determining whether a submitted upgrade command for the existing application is received when the submitted application has the same name as the existing application; and triggering
 5 the predetermined disable command when the submitted upgrade command for the existing application is received, to invoke the end function of each node in the existing application.

[0060] Compared with the previous implementation, in the present implementation, the application management module does not directly upgrade the
 10 existing application when detecting that the submitted application has the same name, and it needs the user's confirmation. After determining, the user can return an upgrade command. The application management module triggers the predetermined disable command only when the submitted upgrade command for the existing application is received, to invoke the end function of each node in the existing application.

15 [0061] In a schematic diagram illustrating prompt information shown in FIG. 3, the application management module can output prompt information used for determining: "Are you sure to upgrade the application?", and can further display details about the existing application that has the same name and details about the submitted application that has the same name.

20 [0062] According to the present implementation, the application management module is reconstructed, so that the application management module can allow the user to submit the application that has the same name, and simplify two commands that need to be submitted manually into one upgrade command. As such, the user needs to submit one upgrade command, so that the problem in the previous
 25 implementation can be avoided, and the problem in the existing technology that a relatively long time is consumed because human participation is needed in an application upgrade process still can be alleviated, thereby reducing the time consumed for the application upgrade.

[0063] The method for upgrading an application described in the present
 30 application can be applied to a JStorm scenario. In other words, the application includes a JStorm application, and correspondingly, the process includes a jvm process.

[0064] Generally, when the application processes input data, the procedure can be divided into a plurality of processing steps, and each processing step can be

considered as one node. Usually, each node corresponds to one process. A node connected to a data source can be referred to as a root node (spout), and other nodes can be referred to as sub-nodes (bolt). In a schematic diagram illustrating nodes shown in FIG. 4, node 1 is a root node, and nodes 2, 3, and 4 are sub-nodes. Each
 5 piece of data is processed in a sequence of nodes.

[0065] In the existing technology, when an old application is disabled, it needs to be manually determined that processing of unprocessed data in the old application is completed. However, manual determining is not necessarily accurate, and determining needs to depend on experience. Therefore, it cannot be ensured that the unprocessed
 10 data in the old application is completely processed.

[0066] To alleviate the problem, FIG. 5 shows a flowchart illustrating a method for disabling an application, according to an implementation of the present application. The present implementation is described from the perspective of an application side to be upgraded, and includes the following steps.

[0067] Step 210: After an application management module invokes an end function of a node, stop a root node from receiving backflow data, and determine the root node as a current node.

[0068] In the present implementation, as described above, the root node is connected to a data source, and when the application is disabled, the root node needs
 20 to be stopped from receiving backflow data from the data source.

[0069] Step 220: Process data in the current node and send the data to a downstream node.

[0070] In the present implementation, unprocessed data in the current node needs to be processed, and the processed data needs to be sent to the downstream node. Data
 25 processed by root node 1 in FIG. 4 needs to be sent to the downstream node, namely, node 2.

[0071] Step 230: Broadcast an ending identifier to all downstream sub-nodes of the current node after completing processing of the data in the current node and receiving ending identifiers broadcast by all upstream nodes.

[0072] In the present implementation, the ending identifier is used to indicate that processing of the data in the node is completed. For example, node 2 broadcasts an ending identifier, and it indicates that processing of data in node 2 is completed.

[0073] After completing data processing, the current node needs to determine whether the ending identifiers broadcast by all the upstream nodes are received, and

the current node broadcasts the ending identifier to all the downstream sub-nodes of the current node after completing processing of the data in the current node and receiving the ending identifiers broadcast by all the upstream nodes.

[0074] Step 240: When the current node has a next node, determine the next node as the current node, and return to perform step 220.

[0075] Step 250: When the current node is the last node, the root node returns execution completion information to the application management module to wait for the application management module to recycle a process that corresponds to each node.

[0076] In the present implementation, after receiving the ending identifiers broadcast by all the upstream nodes, the last node can notify the root node, and then the root node returns the execution completion information to the application management module to wait for the application management module to recycle the process that corresponds to each node.

[0077] According to the present implementation, an application ending procedure is reconstructed, so that any node needs to broadcast an ending identifier to all downstream nodes when completing processing of data in the node and receiving ending identifiers broadcast by all upstream nodes. The root node can return the execution completion information to the application management module only after all the nodes complete data processing, to wait for the application management module to recycle the process that corresponds to each node. Therefore, it can be ensured, by using the ending identifier, that processing of data in each node of the application is completed, to ensure the integrity of data processing and avoid omission that occurs when completion of data processing is determined manually. In addition, because there is no human participation, a time consumed for an application upgrade can be reduced.

[0078] In the present implementation, the application management module and the application upgrade procedure are reconstructed, so that when the application is upgraded, the application management module and the application itself can automatically complete upgrade, and the entire process does not need human participation, thereby reducing the time needed for the application upgrade.

[0079] The method for disabling an application described in the present application can be applied to a JStorm scenario. In other words, the application includes a JStorm application, and correspondingly, the process includes a jym

process.

[0080] Corresponding to the previous implementation of the method for upgrading an application, the present application further provides an implementation of an apparatus for upgrading an application.

5 **[0081]** The implementation of the apparatus for upgrading an application in the present application can be applied to an application management module. The apparatus implementation can be implemented by software, hardware, or a combination of hardware and software. Software implementation is used as an example. As a logical apparatus, the apparatus is formed by reading a corresponding
 10 computer program instruction in a nonvolatile memory to a memory by a processor of a device where the apparatus is located. In terms of hardware, FIG. 6 is a structural diagram illustrating hardware of the device where the apparatus for upgrading an application in the present application is located. In addition to a processor, a network interface, a memory, and a nonvolatile memory shown in FIG. 6, the device where the
 15 apparatus in the present implementation is located usually can further include other hardware based on an actual function of the apparatus for upgrading an application. Details are omitted here for simplicity.

[0082] FIG. 7 is a diagram illustrating modules of an apparatus for upgrading an application, according to an implementation of the present application. The present
 20 implementation is described from the perspective of an application management module side. The apparatus includes an acquisition unit 310, an invocation unit 320, a recycling unit 330, and an initiation unit 340.

[0083] The acquisition unit 310 is configured to obtain an application name of a submitted application.

25 **[0084]** The invocation unit 320 is configured to trigger a predetermined disable command when the submitted application has the same name as an existing application, to invoke an end function of each node in the existing application.

[0085] The recycling unit 330 is configured to recycle a process occupied by each node when receiving execution completion information returned by the existing
 30 application.

[0086] The initiation unit 340 is configured to trigger a predetermined start command when recycling is completed, to start a process that corresponds to each node in the submitted application.

[0087] In an optional implementation, the invocation unit 320 includes the

following: a determining subunit, configured to determine whether a submitted upgrade command for the existing application is received when the submitted application has the same name as the existing application; and an invocation subunit, configured to trigger the predetermined disable command when the submitted upgrade
 5 command for the existing application is received, to invoke the end function of each node in the existing application.

[0088] In an optional implementation, the application includes a JStorm application.

[0089] The process includes a jvm process.

10 **[0090]** Corresponding to the previous implementation of the method for disabling an application, the present application further provides an implementation of an apparatus for disabling an application.

[0091] The implementation of the apparatus for disabling an application in the present application can be applied to an application to be upgraded. The apparatus
 15 implementation can be implemented by software, hardware, or a combination of hardware and software. Software implementation is used as an example. As a logical apparatus, the apparatus is formed by reading a corresponding computer program instruction in a nonvolatile memory to a memory by a processor of a device where the apparatus is located. In terms of hardware, FIG. 8 is a structural diagram illustrating
 20 hardware of the device where the apparatus for disabling an application in the present application is located. In addition to a processor, a network interface, a memory, and a nonvolatile memory shown in FIG. 8, the device where the apparatus in the present implementation is located usually can further include other hardware based on an actual function of the apparatus for disabling an application. Details are omitted here
 25 for simplicity.

[0092] FIG. 9 is a diagram of modules of an apparatus for disabling an application, according to an implementation of the present application. The present implementation is described from the perspective of an application side to be upgraded. The apparatus includes a receiving stopping unit 410, a data processing unit
 30 420, an identifier broadcasting unit 430, a control unit 440, and an information returning unit 450.

[0093] The receiving stopping unit 410 is configured to: after an application management module invokes an end function of a node, stop a root node from receiving backflow data, and determine the root node as a current node.

[0094] The data processing unit 420 is configured to process data in the current node and send the data to a downstream node.

[0095] The identifier broadcasting unit 430 is configured to broadcast an ending identifier to all downstream sub-nodes of the current node after processing of the data in the current node is completed and ending identifiers broadcast by all upstream nodes are received.

[0096] The control unit 440 is configured to: when the current node has a next node, determine the next node as the current node, and then input data of the next node to the data processing unit.

[0097] The information returning unit 450 is configured to: when the current node is the last node, return, by the root node, execution completion information to the application management module to wait for the application management module to recycle a process that corresponds to each node.

[0098] In an optional implementation, the application includes a JStorm application.

[0099] The process includes a jvm process.

[0100] In conclusion, in the implementations of the present application, the application management module (such as a topology manager (TM)) is reconstructed, so that the application management module can allow a user to submit the application that has the same name as the existing application. Therefore, the application management module can automatically upgrade the existing application that has the same name when detecting that the user submits the application that has the same name. As such, the problem in the existing technology that a relatively long time is consumed because human participation is needed in an application upgrade process can be avoided, thereby reducing a time consumed for an application upgrade. In the implementations of the present application, an application ending procedure is further reconstructed, so that any node needs to broadcast an ending identifier to all downstream nodes when completing processing of data in the node and receiving ending identifiers broadcast by all upstream nodes. The root node can return the execution completion information to the application management module only after all the nodes complete data processing, to wait for the application management module to recycle the process that corresponds to each node. Therefore, it can be ensured, by using the ending identifier, that processing of data in each node of the application is completed, to ensure the integrity of data processing and avoid omission

that occurs when completion of data processing is determined manually. In addition, because there is no human participation, the time consumed for the application upgrade can be reduced.

5 **[0101]** For an implementation process of functions and roles of the units in the apparatus, references can be made to an implementation process of corresponding steps in the previous method. Details are omitted here for simplicity.

10 **[0102]** Because an apparatus implementation basically corresponds to the method implementation, for related parts, references can be made to related descriptions in the method implementation. The previously described apparatus implementation is merely an example. The units described as separate parts may or may not be physically separate, and parts displayed as units may or may not be physical units, may be located in one position, or may be distributed on a plurality of network units. Some or all of the modules can be selected based on actual needs to achieve the objectives of the solutions in the present application. A person of ordinary skill in the art can understand and implement the implementations of the present application without creative efforts.

15 **[0103]** A person skilled in the art can easily figure out another implementation solution of the present application after considering the present specification and practicing the disclosed invention here. The present application is intended to cover any variations, functions, or adaptive changes of the present application. These variations, functions, or adaptive changes comply with general principles of the present application, and include common knowledge or a commonly used technical means in the technical field that is not disclosed in the present application. The present specification and the implementations are merely considered as examples. The actual scope and the spirit of the present application are pointed out by the following claims.

20 **[0104]** It should be understood that the present application is not limited to the accurate structures described above and shown in the accompanying drawings, and modifications and changes can be made without departing from the scope of the present application. The scope of the present application is limited only by the appended claims.

CLAIMS

What is claimed is:

1. A computer-implemented method for disabling an application, the computer-implemented method comprising:

invoking, by an application management module, an end function of a node of the application;

stopping a root node of the application from receiving backflow data;

determining the root node as a current node;

processing data in the current node;

sending the data to a downstream node;

completing processing of the data in the current node;

broadcasting an ending identifier to downstream sub-nodes of the current node; and

receiving ending identifiers broadcast by upstream nodes.

2. The computer-implemented method of claim 1, further comprising:

determining whether the current node comprises a next node;

in response to determining that the current node comprises the next node, determining the next node as the current node;

returning to perform the step of processing the data in the current node; and

sending the data to the downstream node.

3. The computer-implemented method of claim 1, further comprising:
determining whether the current node comprises a last node; and
in response to determining that the current node comprises the last node, returning, by the root node, execution completion information to the application management module to wait for the application management module to recycle a process that corresponds to each node.

4. The computer-implemented method of claim 3, further comprising:
triggering a predetermined start command when recycling is completed, to start a process that corresponds to each node in the application.

5. The computer-implemented method of claim 1, wherein the application is configured to concurrently process a plurality of threads, each thread running a corresponding sub-process.

6. The computer-implemented method of claim 1, further comprising:
retrieving an application name of the application submitted for an upgrade;
determining whether the application name corresponds to an existing application; and
in response to determining that the application name corresponds to the existing application, triggering a predetermined disable command, to invoke an end function of each node in the existing application.

7. The computer-implemented method of claim 6, wherein triggering the predetermined disable command comprises:

determining whether a submitted upgrade command for the existing application is received;
and

in response to determining that the submitted upgrade command for the existing application is received, triggering the predetermined disable command to invoke the end function of each node in the existing application.

8. An apparatus for upgrading an application, the apparatus comprising a plurality of modules configured to perform the computer-implemented method of any one of claims 1 to 7.