

TRAFFIC LIMITING METHOD AND SYSTEM, AND STORAGE MEDIUM

TECHNICAL FIELD

[0001] This specification relates to the field of network technologies, and in particular, to a
5 traffic limiting method and system, and a storage medium.

BACKGROUND

[0002] A current Internet service architecture includes a platform that provides a service and
a service user who uses a service.

[0003] To ensure stability and service quality of the platform and prevent the service user from
10 abusing service resources of the platform, the platform can limit an access request of the service
user through traffic limiting. For example, the platform can limit a request rate of an interface that
provides a service, to limit a quantity of requests initiated by the service user through the interface
within a unit time.

[0004] In this way, if the quantity of requests initiated by the service user through the interface
15 within the unit time exceeds a corresponding limit, there may be problems such as a request failure,
an information loss, or limited access because the corresponding limit is exceeded.

[0005] Content in the background is merely information known to the inventor, and neither
means that the information has entered the public domain before the application date of this
disclosure, nor means that the information can become the existing technology of this disclosure.

SUMMARY

[0006] This specification provides a traffic limiting method and system, and a storage medium,
to avoid at least one of the above-mentioned technical problems.

[0007] According to a first aspect, this specification provides a traffic limiting method, applied
to an intermediary platform, and including: obtaining a target task of a user, where the target task
25 is used to represent invocation of a target interface of a third-party platform to perform a target

operation; obtaining traffic limiting information of the target interface; generating a current token for the target task based on the traffic limiting information; and when the current token is a valid token that meets the traffic limiting information, invoking the target interface to execute the target task.

5 **[0008]** According to a second aspect, this specification further provides a traffic limiting system, applied to an intermediary platform, and including: at least one storage medium that stores at least one instruction set for traffic limiting; and at least one processor that is communicatively connected to the at least one storage medium. When the traffic limiting system runs, the at least one processor reads the at least one instruction set, and performs the following operations based
10 on an indication of the at least one instruction set: obtaining a target task of a user, where the target task is used to represent invocation of a target interface of a third-party platform to perform a target operation; obtaining traffic limiting information of the target interface; generating a current token for the target task based on the traffic limiting information; and when the current token is a valid token that meets the traffic limiting information, invoking the target interface to execute the target
15 task.

[0009] According to a third aspect, this specification further provides a computer-readable nonvolatile storage medium. The computer-readable nonvolatile storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the traffic limiting method according to the first aspect of this specification is implemented.

20 **[0010]** It can be learned from the above-mentioned technical solutions that this specification provides the traffic limiting method and system, and the storage medium. The traffic limiting method can be applied to the intermediary platform. When the user needs to perform the target operation on the third-party platform, in consideration of a traffic constraint of the third-party platform, the intermediary platform can determine whether the target operation does not exceed
25 the traffic constraint of the third-party platform, so that when the target operation does not exceed the traffic constraint of the third-party platform, the target interface of the third-party platform is invoked to perform the target operation. In this way, a disadvantage that when the user directly interacts with the third-party platform, if a corresponding request initiated by the user exceeds the traffic constraint of the third-party platform, data of the corresponding request is lost can be
30 avoided. Therefore, reliability of a request of the user for a corresponding operation can be improved, thereby improving user experience. In addition, the intermediary platform invokes the

target interface by using the traffic limiting information and the valid token. This can effectively meet a requirement of invoking the target interface within the traffic limit of the target interface, further avoid data missing of the request, and improve effectiveness and reliability of traffic limiting.

- 5 **[0011]** Other functions of the traffic limiting method and system, and the storage medium provided in this specification are partially listed in the following descriptions. Creative aspects of the traffic limiting method and system, and the storage medium provided in this specification can be fully explained by practice or by using the methods, apparatuses, and combinations described in the following detailed examples.

10 **BRIEF DESCRIPTION OF DRAWINGS**

- [0012]** To describe the technical solutions in the embodiments of this specification more clearly, the following briefly describes the accompanying drawings needed for describing the embodiments. Clearly, the accompanying drawings in the following description show merely some embodiments of this specification, and a person of ordinary skill in the art can still derive other
15 drawings from these accompanying drawings without creative efforts.

[0013] FIG. 1 is a schematic diagram of an application scenario of a traffic limiting method;

[0014] FIG. 2 is a diagram of a hardware structure of a computing device according to an embodiment of this specification;

20 **[0015]** FIG. 3 is a flowchart of a traffic limiting method according to an embodiment of this specification;

[0016] FIG. 4 is a schematic flowchart of constructing a target asynchronous task list according to an embodiment of this specification; and

[0017] FIG. 5 is a schematic diagram of an implementation principle of a traffic limiting method according to this specification.

25 **DESCRIPTION OF EMBODIMENTS**

[0018] Specific application scenarios and requirements of this specification are provided in the following descriptions, so that a person skilled in the art can manufacture and use content of this specification. Various local modifications to the disclosed embodiments are clear to a person

skilled in the art. In addition, the general principles defined herein can be applied to other embodiments and applications without departing from the spirit and scope of this specification. Therefore, this specification is not limited to the shown embodiments, but has a widest scope consistent with the claims.

5 **[0019]** The terms used herein are merely intended to describe specific example embodiments, and impose no limitation. For example, unless otherwise explicitly stated in the context, the singular forms "one", "an", and "the" used herein can include plural forms. When being used in this specification, the terms "include", "comprise", and/or "have" mean existence of an associated integer, step, operation, element, and/or component, but do not preclude existence of one or more
10 other features, integers, steps, operations, elements, components, and/or groups or addition of other features, integers, steps, operations, elements, components, and/or groups to the system/method.

[0020] In consideration of the following descriptions, these and other features of this specification, operations and functions of related components of the structure, and economy of combination and manufacturing of components can be significantly improved. With reference to
15 the accompanying drawings, all of these form a part of this specification. However, it should be clearly understood that the accompanying drawings are merely used for the purpose of illustration and description, and are not intended to limit the scope of this specification. It should be further understood that the accompanying drawings are not drawn to scale.

[0021] The flowchart used in this specification shows operations implemented by a system
20 according to some embodiments of this specification. It should be clearly understood that the operations in the flowchart can be implemented out of sequence. On the contrary, the operations can be implemented reversely or simultaneously. In addition, one or more other operations can be added to the flowchart, and one or more operations can be removed from the flowchart.

[0022] For ease of description, the terms that appear below in this specification are first
25 explained.

[0023] A platform interface usually refers to a group of rules, definitions, and protocols used in a software system or service to interact with an external system or a user. The interface allows the user or a third-party application to interact with a platform by using a predefined protocol and method, to perform operations such as data retrieval, submission, and update.

30 **[0024]** Request frequency limiting (Rate Limiting) means that a platform interface limits a quantity of requests allowed to be received within a unit time, to prevent excessive requests from

causing pressure or abuse to a system. A common limiting manner includes a maximum quantity of requests per second, per minute, or per hour.

[0025] A request frequency limiting rule refers to a constraint condition set by a platform that provides a service for a quantity of requests that can be initiated by an external system within a specific time range. The rule is used to control and manage a frequency at which the system sends a request to the platform that provides a service, to prevent excessive requests from causing overload of the platform that provides a service and ensure system stability and fair use of the platform that provides a service.

[0026] A maximum quantity of requests is a quantity of requests allowed to be sent in a specific time window (for example, per minute, per hour, or per day) in a request frequency limiting rule.

[0027] A time window refers to a time period on which a traffic limiting rule is based, for example, "per minute", "per hour", or "per day".

[0028] Traffic limiting refers to a mechanism for controlling a quantity of access times, and is used to manage and limit a frequency of access to a specific resource or service.

[0029] A token is an identifier used to control request execution, and is generated based on a traffic limit of a platform interface. The token is used to ensure that a task is executed within a request frequency limiting range, so as to avoid a request failure or loss caused because the limit is exceeded.

[0030] An application scenario of this specification is described below.

[0031] The traffic limiting method and system provided in this specification can be applied to a scenario in which a third-party platform performs traffic limiting for a service user. In the scenario, when the service user needs to use a service provided by the third-party platform, an interface of the third-party platform can be invoked through an intermediary platform (for example, the traffic limiting system provided in this specification) for implementation. The third-party platform includes but is not limited to a takeout platform, an online shopping platform, a multimedia platform, or the like.

[0032] For example, a scenario in which the third-party platform is a takeout platform is used as an example.

[0033] The service user is a user registered with the takeout platform, and is usually a merchant that provides a takeout service.

[0034] In an example, the merchant may need to synchronously remove a product on the

takeout platform because the product is sold out. The merchant may frequently initiate a request to the intermediary platform for synchronously removing the product on the takeout platform. According to the traffic limiting method provided in this specification, the intermediary platform can avoid a loss of a request of the merchant.

5 **[0035]** For another example, there are a plurality of merchants, and each of the plurality of merchants initiates, through the intermediary platform, a request for synchronously removing a product on the takeout platform. According to the traffic limiting method provided in this specification, the intermediary platform can avoid a loss of requests of at least some merchants.

10 **[0036]** According to the traffic limiting method and system, and the storage medium provided in this solution, a corresponding operation is performed on a third-party platform by an intermediary platform based on a request initiated by a merchant, to avoid a problem of a request loss or traffic limiting caused because the request exceeds a traffic limit of the third-party platform.

15 **[0037]** It should be noted that the above-mentioned scenario is merely used for an example to describe an application scenario to which the technical solutions provided in this specification may be applicable, and cannot be understood as a limitation on the application scenario of the technical solutions provided in this specification. A person skilled in the art should understand that when the traffic limiting method and system provided in this specification are applied to another use scenario, an implementation and technical effects thereof are similar.

20 **[0038]** FIG. 1 is a schematic diagram of an application scenario of a traffic limiting method. The application scenario 100 can include an intermediary platform 101, a user 102, a client 103, and a third-party platform 104.

25 **[0039]** The intermediary platform 101 can be a platform for the user 102 to manage a resource of the user 102 on the third-party platform 104. The intermediary platform 101 can be a server, a computing device, a processor, or the like. If the intermediary platform 101 is a server, the intermediary platform 101 can be a cloud server or a local server.

30 **[0040]** The traffic limiting method provided in this specification can be performed on the intermediary platform 101. In this case, the intermediary platform 101 can store data or instructions for performing the traffic limiting method described in this specification, and can execute or be configured to execute the data or the instructions. In some embodiments, the intermediary platform 101 can include a hardware device with a data information processing function and a necessary program needed to drive the hardware device to work.

[0041] The user 102 can be a takeout merchant, an e-commerce merchant, or any natural person or enterprise registered with the third-party platform. The user 102 manages the resource of the user 102 on the third-party platform 104 through the intermediary platform 101, for example, updates product information, processes an order, or performs inventory management on the third-party platform 104. Specifically, the user 102 sends a corresponding instruction to the intermediary platform 101 through the client 103, to indicate, by using the user instruction, the intermediary platform 101 to schedule a corresponding interface of the third-party platform 104, so as to perform a corresponding operation.

[0042] The client 103 can be an electronic device that provides an interaction function to the user 102. For example, the client 103 can provide an interaction interface to the user 102, and the user 102 can perform an interaction operation in the interaction interface. In some embodiments, in response to detecting a user instruction triggered by the user 102, the client 103 sends the user instruction to the intermediary platform 101, so that the intermediary platform 101 performs the traffic limiting method described in this specification.

[0043] For example, if the third-party platform 104 is a takeout platform, the user 102 can be a merchant having a store registered with the third-party platform 104, and the client 103 usually refers to a device used by the merchant to manage the store of the merchant on the takeout platform. Through the client 103, a user instruction can be initiated to the intermediary platform 101, to manage a store of the user on the takeout platform by using the user instruction.

[0044] Specifically, as a takeout merchant, the user 102 can send a user instruction such as product removal, inventory update, or order processing to the intermediary platform 101 through the client 103. In this scenario, the client 103 can be a POS machine.

[0045] In some embodiments, the client 103 can include a mobile device, a tablet computer, a laptop computer, a device built in a motor vehicle or similar content, or any combination thereof. In some embodiments, the mobile device can include a smart home device, a smart mobile device, a virtual reality device, an augmented reality device or a similar device, or any combination thereof. In some embodiments, the smart home apparatus can include a smart television, a desktop computer, or any combination. In some embodiments, the smart mobile device can include a smartphone, a personal digital assistant, a game device, a navigation device, or any combination thereof. In some embodiments, the apparatus built in the motor vehicle can include a vehicle-mounted computer, a vehicle-mounted television, or the like.

[0046] In some embodiments, one or more applications (app) can be installed on the client 103. The app can provide a capability and an interface for interacting with the outside through a network to the user 102. The app includes but is not limited to a web browser app program, a search app program, a chat app program, a shopping app program, a video app program, a financial app program, an instant messaging tool, a mailbox client, social platform software, or the like.

[0047] The third-party platform 104 can be a platform that provides a service to the user 102, for example, a takeout platform, a shopping platform, or a multimedia platform. The third-party platform 104 can include a plurality of interfaces, to provide different services through different interfaces. For example, the third-party platform 104 is still a takeout platform. As a takeout merchant, the user 102 can invoke, through the intermediary platform 101, the interfaces of the third-party platform 104 to perform operations such as store management, order processing, and product launching/removal. Specifically, for the third-party platform 104, an interface can be used for product launching, and another interface can be used for product removal.

[0048] The intermediary platform 101 and the client 103 and the intermediary platform 101 and the third-party platform 104 can be separately communicatively connected through a network. The network can facilitate exchange of information or data. In some embodiments, the network can be any type of wired or wireless network, or a combination thereof. For example, the network can include a cable network, a wired network, an optical fiber network, a telecommunications network, an intranet, the Internet, a local area network (LAN), a wide area network (WAN), a wireless local area network (WLAN), a metropolitan area network (MAN), a public switched telephone network (PSTN), a Bluetooth network, a ZigBee network, a near field communication (NFC) network, or a similar network. In some embodiments, the network can include one or more network access points. For example, the network can include a wired or wireless network access point, for example, a base station or an Internet switching point. Through the access point, one or more components of the intermediary platform 101 and the client 103 can be connected to the network to exchange data or information.

[0049] It can be understood that FIG. 1 and the above-mentioned example are merely used as examples to describe an application scenario to which the traffic limiting method provided in this specification may be applicable, and cannot be understood as a limitation on the application scenario. For example, the intermediary platform 101 can be communicatively connected to one client 103, as shown in FIG. 1. In some other embodiments, the intermediary platform 101 can be

communicatively connected to a plurality of clients 103, and obtain user instructions respectively sent by the clients for unified scheduling. Similarly, the intermediary platform 101 can be communicatively connected to one third-party platform 104, as shown in FIG. 1. In some other embodiments, the intermediary platform 101 can be communicatively connected to a plurality of
5 third-party platforms 104, to manage user resources on the third-party platforms 104.

[0050] FIG. 2 is a diagram of a hardware structure of a computing device 200 according to an embodiment of this specification. The computing device 200 can serve as the intermediary platform 101 in FIG. 1 to perform the traffic limiting method described in this specification.

[0051] As shown in FIG. 2, the computing device 200 can include at least one storage medium
10 230 and at least one processor 220. In some embodiments, the computing device 200 can further include a communication port 250 and an internal communication bus 210. The computing device 200 can further include an I/O component 260.

[0052] The internal communication bus 210 can be connected to different system components. For example, the internal communication bus 210 can be connected to the storage medium 230,
15 the processor 220, the communication port 250, the I/O component 260, and the like.

[0053] The I/O component 260 supports an input/output between the computing device 200 and another component.

[0054] The communication port 250 is used for data communication between the computing device 200 and the outside. For example, the communication port 250 can be used for data
20 communication between the computing device 200 and a network. The communication port 250 can be a wired communication port, or can be a wireless communication port.

[0055] The storage medium 230 can include a data storage apparatus. The data storage apparatus can be a non-transitory storage medium, or can be a transitory storage medium. For example, the data storage apparatus can include one or more of a disk 232, a read-only storage medium (ROM) 234, or a random access storage medium (RAM) 236. The storage medium 230
25 further includes at least one instruction set stored in the data storage apparatus. The instruction set can include computer program code. The computer program code can include a program, a routine, an object, a component, a data structure, a process, a module, and the like.

[0056] The at least one processor 220 can be communicatively connected to the at least one
30 storage medium 230. When the computing device 200 runs, the at least one processor 220 reads the at least one instruction set, and performs, based on an indication of the at least one instruction

set, the traffic limiting method provided in this specification. The processor 220 can perform the steps included in the traffic limiting method. The processor 220 can be in a form of one or more processors. In some embodiments, the processor 220 can include one or more hardware processors, for example, a microcontroller, a microprocessor, a reduced instruction set computer (RISC), an application-specific integrated circuit (ASIC), an application-specific instruction set processor (ASIP), a central processing unit (CPU), a graphics processing unit (GPU), a physical processing unit (PPU), a microcontroller unit, a digital signal processor (DSP), a field programmable gate array (FPGA), an advanced RISC machine (ARM), a programmable logic device (PLD), any circuit, processor, or the like that can perform one or more functions, or any combination thereof.

10 **[0057]** To merely describe a problem, only one processor 220 is shown in the computing device 200 in the accompanying drawings. However, it should be noted that the computing device 200 in this specification can further include a plurality of processors. Therefore, the operations and/or method steps disclosed in this specification can be performed by one processor, or can be jointly performed by a plurality of processors. For example, if the processor 220 in the computing device 200 described in this specification performs step A and step B, it should be understood as that step A and step B can be jointly or separately performed by two different processors 220 (for example, a first processor performs step A and a second processor performs step B, or a first processor and a second processor jointly perform step A and step B).

[0058] FIG. 3 is a flowchart of a traffic limiting method P300 according to an embodiment of this specification. As shown in FIG. 3, the method P300 can include the following steps S320 to S350.

[0059] S320: Obtain a target task of a user, where the target task is used to represent invocation of a target interface of a third-party platform to perform a target operation.

[0060] For example, the target interface is an interface corresponding to the target operation. For example, the third-party platform can include a plurality of interfaces, and different interfaces correspond to different operations.

[0061] With reference to the above-mentioned analysis, the third-party platform can be a takeout platform, and the user can be a takeout merchant registered with the third-party platform. The target operation can be an operation of removing a product in a store corresponding to the takeout merchant on the takeout platform, and the target interface is an interface for requesting to perform the operation of removing the product on the takeout platform.

- [0062]** In some embodiments, the target task is obtained from a target asynchronous task list. The target asynchronous task list is a structure used to store and process a task executed in background, and is used on an intermediary platform to manage and schedule a target task corresponding to a user instruction sent by the user. For example, when the user sends an instruction, the intermediary platform can construct a corresponding target task based on the user instruction, and add the target task to the target asynchronous task list.
- [0063]** FIG. 4 is a schematic flowchart of constructing a target asynchronous task list according to an embodiment of this specification. As shown in FIG. 4, the constructing a target asynchronous task list includes the following steps S401 to S403.
- 10 **[0064]** S401: Obtain a user instruction initiated by the user, where the user instruction is used to indicate to perform a corresponding operation on the third-party platform.
- [0065]** For example, the corresponding operation can be initiating a single task, for example, updating a product status or modifying order information; or can be initiating a batch of tasks, for example, synchronizing merchant data.
- 15 **[0066]** To help a reader further understand an implementation principle of the traffic limiting method provided in this specification, this embodiment is described with reference to FIG. 5 and by using an application scenario of the takeout platform as an example. The user is a merchant, a client is a POS machine, the intermediary platform is an intermediate server, and the third-party platform is a takeout platform.
- 20 **[0067]** As shown in FIG. 5, in step 1, the merchant initiates an instruction (that is, a user instruction) for managing the takeout platform to the intermediate server through the POS machine. The user instruction can be used to indicate the intermediate server to perform an operation on related information of a store of the merchant on the takeout platform, to correspondingly manage the store of the merchant.
- 25 **[0068]** For example, the user instruction can be "Remove a product X on the takeout platform". That is, the user instruction is used to indicate the intermediate server to perform an operation of removing the product X in the store of the merchant on the takeout platform on the takeout platform.
- [0069]** Correspondingly, the intermediate server receives the user instruction sent by the merchant.
- 30 **[0070]** S402: Query an initial task list to determine whether the initial asynchronous task list includes a task corresponding to the user instruction, to obtain a query result.

- [0071] The initial asynchronous task list can be understood as a queue that stores a task that has not been executed, and a task in the queue may be generated based on a previous user instruction or an operation spontaneously created by a system.
- [0072] It should be understood that the initial asynchronous task list corresponds to the target asynchronous task list. "Initial" and "target" are mainly used to distinguish between asynchronous task lists in different phases, and cannot be understood as a limitation on the asynchronous task list. For example, the initial asynchronous task list can be understood as a list before the target task corresponding to the user instruction is added to the list, and the target asynchronous task list can be understood as a list after the target task is added to the initial asynchronous task list.
- [0073] Still with reference to FIG. 5 and the above-mentioned example, after receiving the user instruction, the intermediate server can perform step 2: Query whether there is currently a request that is being processed and that corresponds to the user instruction.
- [0074] S403: If the query result is that the initial asynchronous task list includes no task corresponding to the user instruction, add the target task corresponding to the user instruction to the initial asynchronous task list, to obtain the target asynchronous task list that includes the target task.
- [0075] The query result may include two cases. A case (1) is described in S403, and the initial asynchronous task list includes no task corresponding to the user instruction. In this case, the intermediary platform can add the target task corresponding to the user instruction to the initial asynchronous task list, to update the initial asynchronous task list to obtain the target asynchronous task list.
- [0076] Still with reference to FIG. 5 and the above-mentioned example, for the case (1), the intermediate server can perform step 3: If there is no request that is being processed and that corresponds to the user instruction, add the target task corresponding to the user instruction to an asynchronous processing list (that is, the initial asynchronous task list).
- [0077] For example, the user instruction can be "Remove a product X on the takeout platform A". If the query result is no, it indicates that the merchant currently does not have a same instruction in the intermediate server for execution. Therefore, the intermediate server adds the target task corresponding to the user instruction and obtains the target asynchronous task list, to obtain an operation corresponding to the target task from the target asynchronous task list and perform the operation.

[0078] As shown in FIG. 5, the intermediate server can perform step 4: Obtain the target task from the target asynchronous task list. For example, the intermediate server can obtain the target task from the target asynchronous task list in a "first in first out" manner. Alternatively, the intermediate server can obtain the target task from the target asynchronous task list in a random extraction manner.

[0079] With reference to the above-mentioned analysis for S401 to S403, it can be learned that in this embodiment, the intermediary platform performs a corresponding operation by constructing an asynchronous task list, so that requests of the user can be executed in an orderly manner, and another operation of the user is not affected. In addition, repeated execution of a same request can be avoided through query, and consumption of traffic of the third-party platform can be reduced, thereby improving utilization of traffic resources.

[0080] As shown in FIG. 5, in some embodiments, for the case (1), the intermediate server can perform step 5: Feed back prompt information to the merchant through the POS machine, to prompt the merchant that the request is accepted and please wait.

[0081] It can be understood that in addition to the case (1), the query result can be a case (2): The query result is yes.

[0082] For example, still with reference to the above-mentioned example, the initial asynchronous task list may already include a task indicating that the operation of "removing the product X" is performed on the takeout platform A, which indicates that the merchant previously sends a same user instruction to the intermediate server, but the instruction is not completed.

[0083] For the case (2), as shown in FIG. 5, the intermediate server can perform step 6: Output a prompt message to the user to prompt the user to try again later.

[0084] That is, if the query result is yes, that is, the initial asynchronous task list includes the task corresponding to the current user instruction, the intermediate server performs a prompt operation. In this case, the merchant may have sent a same request, and the request is currently waiting for execution by the intermediate server. To avoid repeated processing of a task or a resource waste, the intermediate server outputs a prompt message to notify the user to try again later, so as to avoid repeatedly submitting a same task in a short time. In addition, a waste that is of resources of the intermediate server and that is caused due to repeated processing can be avoided.

That is, this prompt mechanism facilitates resource utilization, avoids overloading caused due to a repeated request, and can avoid excessive occupation that is of resources of the intermediate

server and that is caused due to a repeated operation of the merchant.

[0085] S330: Obtain traffic limiting information of the target interface.

[0086] For example, the traffic limiting information refers to a constraint rule for a request frequency or quantity of the target interface. In some embodiments, the traffic limiting information can be used to represent a maximum quantity of invocation times of the target interface in a preset time window.

[0087] That is, the traffic limiting information can represent a quantity of requests that can be sent by the intermediary platform to the target interface within a unit time. A range of the unit time can be in a unit such as second, minute, or hour.

[0088] With reference to the above-mentioned analysis, the third-party platform can be a takeout platform, and the target interface can be an interface for managing a resource of the merchant on the takeout platform by the intermediary platform, for example, performing an operation such as launching/removing a product on the takeout platform. The traffic limiting information can include a maximum quantity of product launching requests that can be received by the target interface per second. The traffic limiting information can further include a rule for performing processing after a quantity of product launching requests received in a specific second exceeds a limit.

[0089] For example, still with reference to FIG. 5, after the target task is obtained, the intermediate server can perform step 7: Obtain traffic limiting information of the target interface.

That is, the intermediate server obtains the traffic limiting information of the target interface that is on the takeout platform and that is used to process the target task.

[0090] A manner of obtaining the traffic limiting information is not limited in this embodiment. For example, the intermediary platform can obtain the traffic limiting information by parsing a response header of the target interface, querying an interface document, or directly accessing a dedicated query interface.

[0091] S340: Generate a current token for the target task based on the traffic limiting information.

[0092] For example, the intermediary platform can parse the traffic limiting information, for example, extract a time window size of the target interface and a maximum quantity of requests that can be received by the target interface in a time window, to generate a token (that is, the current token) for the target task.

[0093] That is, this step can be understood as follows: When obtaining the traffic limiting information of the target interface, the intermediary platform generates, based on the traffic limiting information, the current token for the target task for which a request needs to be made to the target interface for implementation. As shown in FIG. 5, when the traffic limiting information is obtained, the intermediate server can perform step 8: Generate a current token based on the traffic limiting information.

[0094] In some embodiments, S340 can include the following steps S341 and S342.

[0095] S341: Determine a current traffic window for the target task based on the traffic limiting information.

[0096] For example, the intermediary platform determines, based on the traffic limiting information, for example, a unit time length of the target interface, a traffic window in which a current time is located. For example, if the traffic window of the target interface is 1 minute and a maximum quantity of requests allowed per minute is 100, the intermediary platform 100 considers every 1 minute as a time window.

[0097] Based on the current time, the intermediary platform uses a next complete time window starting from a current moment as a tentative traffic window. For example, if a current time point is an end moment of a time window (and is a start moment of a next time window), the intermediary platform considers the next time window that is about to start as a tentative traffic window. If the current time point is within a time window, the intermediary platform considers a next time window for the time window as a tentative traffic window.

[0098] After determining the tentative traffic window, the intermediary platform can determine, based on a historical request record in the tentative traffic window, whether a request can continue to be sent to the target interface in the time window. If an available quantity of requests in the tentative traffic window is not exhausted, the intermediary platform determines the tentative traffic window as the current traffic window for the target task. If a quantity of remaining requests in the current tentative traffic window is insufficient, the intermediary platform modifies the tentative traffic window to a next time window, or takes other measures to re-determine the current traffic window for the target task to avoid exceeding a traffic limit.

[0099] For example, in a scenario in which the third-party platform is still a takeout platform, it is assumed that the target task is to remove a dish X on the takeout platform A, and the corresponding traffic limiting information of the target interface is a maximum quantity, that is,

100, of requests allowed every 1 minute. It is assumed that the intermediary platform considers a 0th second to a 59th second in each minute as a time window. If the current time is 12:01:00, which is a start moment of a time window "12:01:00–12:01:59", the intermediary platform uses the time window as a tentative traffic window. If the current system time is 12:01:25, which is a middle
5 moment of a time window "12:01:00–12:01:59", the intermediary platform uses a next time window (that is, 12:02:00–12:02:59) of the time window as a tentative traffic window. The intermediary platform then checks a historical request record before the tentative traffic window and compares the historical request record with the traffic limiting information of the target interface. For example, if the historical request record shows that 30 requests for
10 launching/removing a dish on the takeout platform A are received, because the maximum quantity, that is, 100, of requests of the target interface in the time window is not exhausted, the intermediary platform determines the tentative traffic window as the current traffic window for the target task. In this window, for the target task, a request can be sent to a product removal interface of the takeout platform A without exceeding the traffic limit of the target interface.

15 **[0100]** S342: Generate the current token based on historical token generation information in the current traffic window.

[0101] For example, when a traffic window (that is, the current traffic window) to which the target task belongs is determined, the intermediary platform can obtain information about a historically generated token (that is, the historical token generation information) in the current
20 traffic window. Then, the intermediary platform can generate a token (that is, the current token) corresponding to the target task based on the historical token generation information.

[0102] Based on the above-mentioned analysis for S341 and S342, it can be learned that in this embodiment, the intermediary platform divides the time into consecutive time windows based on the current time and the traffic limiting information of the target interface, to facilitate traffic
25 limiting management. The intermediary platform can dynamically adapt to the traffic limit of the target interface based on historical information and the like, properly arrange the traffic window for the target task, and ensure that a quantity of request invocations in each time window complies with a traffic limiting rule, to ensure effective running of the entire traffic limiting system.

[0103] In some embodiments, the historical token generation information includes a quantity
30 of historical tokens generated in the current traffic window; and S342 can include the following steps S3421 and S3422.

- [0104] S3421: Determine a current sequence number of the current token based on the quantity of historical tokens.
- [0105] The current sequence number can be understood as a number corresponding to the current token, for example, indicates a quantity of generated requests.
- 5 [0106] For example, the intermediary platform can retrieve and review, based on the current time, a quantity of tokens used or generated in the current traffic window. Each token represents a valid request opportunity. If a specific quantity of tokens is generated in the current traffic window, the intermediary platform calculates, based on the historical data, a sequence number of a token that should be generated next. When generating the current token, the intermediary platform first
- 10 determines, by using the historical token generation information, the quantity of tokens used in the current traffic window, and then infers, based on the information, the sequence number of the token that should be currently generated.
- [0107] For example, if 80 requests are sent in the current traffic window, and the maximum quantity of requests of the target interface is 100, the current sequence number is 81, and the
- 15 sequence number of the current token generated by the intermediary platform is 81.
- [0108] S3422: Generate a token identifier of the current token based on the current sequence number, an interface identifier of the target interface, and the current time, where the current token includes the token identifier.
- [0109] For example, as shown in FIG. 5, the current token includes the token identifier, and
- 20 the token identifier includes the current sequence number, the interface identifier, and the current time.
- [0110] For example, the token identifier includes the current sequence number of the token, the interface identifier of the target interface, and the current time. The information makes each token traceable in the system. The interface identifier of the target interface is an identifier used to
- 25 uniquely refer to a target interface on the intermediary platform. The current time can be represented by a current timestamp. The token identifier can uniquely identify the current token on the intermediary platform.
- [0111] The scenario in which the third-party platform is a takeout platform is still used as an example. The merchant initiates a plurality of requests through the intermediate server, and wants
- 30 to update a dish inventory on the takeout platform A. The intermediate server can calculate the sequence number of the next generated current token based on the historical data. Then, the

intermediate server can generate a unique token identifier for the current token. The identifier includes the sequence number (for example, 81) of the current token, the identifier of the target interface (for example, a unique ID of the takeout platform A), and the current timestamp (for example, 12:05:30). This token identifier ensures that each token can be uniquely tracked and managed to avoid exceeding a request limit of the platform interface.

[0112] Based on the above-mentioned analysis for S3421 and S3422, it can be learned that the intermediary platform can generate the current token based on the traffic limiting information of the target interface for controlling request sending and traffic management. The process of generating the current token includes determining the sequence number based on the quantity of historical tokens, and generating the unique token identifier with reference to the identifier of the target interface and the timestamp, to ensure that each token is unique and traceable on the intermediary platform, so that the token can be used for execution determining of a subsequent request. Through this procedure, the intermediary platform can ensure that the request is sent in strict compliance with scheduling.

[0113] S350: When the current token is a valid token that meets the traffic limiting information, invoke the target interface to execute the target task.

[0114] For example, if the current token meets the traffic limiting information, that is, the current token represents that invoking the target interface to execute the target task does not exceed the traffic limit of the target interface, it indicates that the current token is a valid token.

[0115] Correspondingly, the intermediary platform can directly invoke the target interface to execute the target task by invoking the target interface.

[0116] Still with reference to the above-mentioned example in which the third-party platform is a takeout platform and the target operation is a product launching operation, if the current token is a valid token, that is, a quantity of times the target interface is invoked to perform the launching operation does not reach the maximum quantity of invocation times, the intermediary platform can directly invoke the target interface, to complete the product launching operation on the takeout platform by invoking the target interface.

[0117] For example, still with reference to FIG. 5, it can be learned that the intermediate server can perform step 9: If the current token is a valid token, invoke the target interface to execute the target task.

[0118] It can be learned from the above-mentioned analysis for S320 to S350 that in this

embodiment, when the user needs to perform the target operation on the third-party platform, in consideration of a traffic constraint of the third-party platform, the intermediary platform can determine whether the target operation does not exceed the traffic constraint of the third-party platform, so that when the target operation does not exceed the traffic constraint of the third-party platform, the target interface of the third-party platform is invoked to perform the target operation. In this way, a disadvantage that when the user directly interacts with the third-party platform, if a corresponding request initiated by the user exceeds the traffic constraint of the third-party platform, data of the corresponding request is lost can be avoided. Therefore, reliability of a request of the user for a corresponding operation can be improved, thereby improving user experience. In addition, the intermediary platform invokes the target interface by using the traffic limiting information and the valid token. This can effectively meet a requirement of invoking the target interface within the traffic limit of the target interface, further avoid data missing of the request, and improve effectiveness and reliability of traffic limiting.

[0119] In some embodiments, the traffic limiting information is used to represent a maximum quantity of invocation times of the target interface in a preset time window. If the maximum quantity of invocation times is greater than the current sequence number, the intermediary platform determines that the current token is a valid token.

[0120] For example, the intermediary platform can determine validity of the token by using the traffic limiting information and the token identifier. Token validity represents whether the currently generated token meets a traffic limiting condition, that is, whether the target interface can be invoked at the current time for the target task that has the token when the traffic limiting rule is met. If the token for the target task is valid, it indicates that execution of the target task at the current time does not violate the traffic limiting rule of the target interface. In addition, if the token for the target task is invalid, it indicates that the intermediary platform completely allocates invocation permission of the target interface to another task in the current time window. In this case, execution of the target task may violate the traffic limiting rule of the target interface.

[0121] In this embodiment, the traffic limiting information is used to represent the maximum quantity of invocation times of the target interface in the preset time window, and the intermediary platform can determine, based on the traffic limiting information, whether the currently generated token is valid.

[0122] For example, the intermediary platform can check the generated token identifier of the

current token, and compare the current sequence number with the maximum quantity of invocation times in the traffic limiting information of the target interface. If the maximum quantity of invocation times is greater than the current sequence number, it is determined that the current token is a valid token.

5 **[0123]** With reference to the above-mentioned example, the target task can represent an operation of removing a product on the takeout platform, the target interface is an interface for performing the operation of removing the product on the takeout platform, and the traffic limiting rule of the target interface can be that a maximum quantity of invocation times per minute is 100. In this example, in each minute, the intermediary platform can arrange invocation of the target
10 interface for tasks that are for removing products and that have token sequence numbers 0–99, that is, all tokens whose token sequence numbers are less than 100 are valid. Therefore, if the sequence number of the token for the target task is 100, the token is invalid because this indicates that 100 other tasks for removing products have valid tokens in this minute. If the target task is executed in this case, the target interface may be invoked for a maximum of 101 times in this minute, that is,
15 the traffic limiting rule of the target interface may be violated.

[0124] Based on the above-mentioned analysis, in this embodiment, the intermediary platform can more simply and directly determine token validity by making a comparison with a number value of the current sequence number of the token based on a maximum quantity limit of invocation times in a time window.

20 **[0125]** In some other embodiments, if the quantity of invocation times is less than the current sequence, the intermediary platform adds the target task to the target asynchronous task list for re-execution.

[0126] If the current sequence number is greater than the maximum quantity of invocation times, it indicates that the quantity of requests in the current traffic window reaches an upper limit,
25 the current token is invalid, and the intermediary platform cannot immediately execute the request. In this case, the intermediary platform can re-add the current target task to the asynchronous task list, so that the target task waits for next start, and a retry is made for re-execution. A retry response time is relatively short, and a time interval from determining that the token is valid to re-adding the current target task to the asynchronous task list is relatively short, which facilitates efficient
30 processing of the request by the intermediary platform.

[0127] In some embodiments, an upper retry time limit is set for the target task, and a target

task that exceeds the upper retry time limit is no longer added to the asynchronous task list. For example, if the retry response time for the target task is 1s, and the upper retry time limit is set to one hour, when the target task is retried for one hour, it can be considered that the retry of the task is invalid consumption of system resources, and therefore the task is stopped.

5 **[0128]** In some embodiments, after the target task is completed on the third-party platform, a corresponding execution result can be fed back to the intermediary platform. Correspondingly, the intermediary platform updates a task status of the target task in response to the execution result of the target task.

10 **[0129]** For example, in a process in which the target task is executed, the intermediary platform continuously tracks a response of the third-party platform, to ensure that the request is successfully processed. Specifically, after the target task is processed by using the third-party platform, the intermediary platform receives feedback from the third-party platform, to determine whether the target task is successfully executed.

15 **[0130]** For example, if the target task is successfully executed, the intermediary platform can mark the status of the task as "completed" or "successful", and record execution-related details such as an execution time and response content for the client to query. If the target task fails to be executed, the intermediary platform can update the task status to "failed" or "abnormal" according to a preset error processing strategy, and record error information, a failure cause, and the like. The intermediary platform can further perform error processing, for example, starting a retry
20 mechanism or notifying the user of a failure result.

25 **[0131]** For example, with reference to the above-mentioned example and FIG. 5, the takeout platform can perform step 10: Feed back an execution result to the intermediate server. Correspondingly, the intermediate server receives the execution result fed back by the takeout platform. On this basis, the intermediate server can perform step 11: Update a task status of the target task.

30 **[0132]** Specifically, when the merchant manages the store on the takeout platform A through the intermediate server, the target operation can be updating a dish inventory. The intermediate server can send a request to the third-party takeout platform A and track an execution result of the request. If the inventory is successfully updated, the intermediate server can update the task status to "completed", and record related information such as an inventory update time, an updated inventory quantity, and other content. If the request fails (for example, there is an interface timeout

or error response), the intermediate server can update the task status to "failed" or "abnormal", and record an error cause.

[0133] With reference to the above-mentioned analysis, it can be learned that in this embodiment, the intermediary platform can clearly record a processing status of the task by recording the execution result of the target task, to ensure that the user can subsequently obtain feedback indicating whether the task is successfully completed and other necessary execution feedback information. In addition, recording of the task status helps the platform track an entire process of task execution, to facilitate subsequent analysis and debugging. If the task fails to be executed, the intermediary platform can record the error cause, retry or perform other processing according to the preset strategy, to avoid impact of a task failure on overall operation.

[0134] In some embodiments, the traffic limiting method provided in this specification further includes: providing feedback based on an updated task status in response to a polling request for the target task.

[0135] For example, after the intermediary platform updates the task status of the target task, when the user sends the polling request to the intermediary platform through the client, the intermediary platform checks a latest status of the target task, and returns corresponding feedback information based on the current status of the task.

[0136] If the target task is successfully completed, the intermediary platform can feed back task success information to the user. The task success information can include a detailed task execution result, for example, response content returned by an interface invocation and an execution timestamp.

[0137] If the target task is not successfully executed or an error occurs, the intermediary platform can notify the user of task failure information in the feedback. The task failure information can include a failure cause, a related error code, a detailed description, and the like. For example, if the task is not successfully executed due to traffic limiting, the intermediary platform can prompt the user to try again later. If the target task is being executed, the intermediary platform returns information indicating that the task is in progress. The information indicating that the task is in progress can include progress of the task, for example, a completed step and a quantity of remaining tasks.

[0138] With reference to the above-mentioned example and FIG. 5, the merchant can perform step 12: Send a polling request to the intermediate server through the POS machine.

Correspondingly, the intermediate server receives the polling request sent by the POS machine. The intermediate server performs step 13: Send polling feedback determined based on an updated task status to the POS machine.

[0139] For example, the merchant requests, through the POS machine, the intermediate server to update the dish inventory on the takeout platform. The intermediate server has updated the task status based on the feedback of the takeout platform. When receiving the polling request, the intermediate server can provide feedback based on the latest status of the task. If the inventory update task is successfully completed, the intermediate server feeds back "task success" information. The information can include a specific updated inventory quantity and an update time. If the task is not successfully completed due to traffic limiting or another reason, the intermediate server feeds back "task failure" information, or can notify the merchant of the failure cause, for example, "The interface request is limited. Please try again later". If the task is still being executed, the intermediate server returns "task in progress" information, and notifies the merchant of current progress, for example, "updated 50% of inventory items", so that the merchant learns of an execution status and remaining workload of the task, thereby improving transparency of task management and operation experience of the merchant.

[0140] With reference to the above-mentioned analysis, it can be learned that in this embodiment, the intermediary platform can feed back the latest status of the target task in real time in response to the polling request of the user or the client, so that the user can learn of the task progress in real time. Through the operation, after the task ends, information can be synchronized with the user and the client in real time to feed back the execution result.

[0141] According to another aspect of this specification, a computer-readable non-transitory storage medium is provided and stores at least one instruction set for traffic limiting. When the at least one instruction set is executed by a processor, the at least one instruction set instructs the processor to implement the steps in the traffic limiting method P300 in this specification. In some possible implementations, various aspects of this specification can be further implemented in a form of a program product, including program code. When the program product runs on the computing device 200, the program code is used to enable the computing device 200 to perform the steps in the traffic limiting method P300 described in this specification. The program product configured to implement the above-mentioned method can use a portable compact disc read-only memory (CD-ROM) that includes program code, and can run on the computing device 200.

However, the program product in this specification is not limited thereto. In this specification, a readable storage medium can be any tangible medium that includes or stores a program, and the program can be used by or in combination with an instruction execution system. The program product can be any combination of one or more readable media. The readable medium can be a readable signal medium or a readable storage medium. For example, the readable storage medium can be but is not limited to an electrical system, apparatus, or device, a magnetic system, apparatus, or device, an optical system, apparatus, or device, an electromagnetic system, apparatus, or device, an infrared system, apparatus, or device, or a semiconductor system, apparatus, or device, or any combination thereof. More specific examples of the readable storage medium include a portable disk, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any proper combination thereof. The computer-readable storage medium can include a data signal propagated in a baseband as a part of a carrier, and carries readable program code. The propagated data signal can be in a plurality of forms, and includes but is not limited to an electromagnetic signal, an optical signal, or any proper combination thereof. The readable storage medium can alternatively be any readable medium other than the readable storage medium, and the readable medium can send, propagate, or transmit a program used by or in combination with an instruction execution system, apparatus, or device. The program code included in the readable storage medium can be transmitted by using any proper medium, including but not limited to a wireless medium, a wired medium, an optical cable, an RF medium, or any proper combination thereof. Program code for performing the operations in this specification can be written in any combination of one or more program design languages. The program design language includes an object-oriented program design language, for example, Java or C++, and further includes a conventional procedural program design language, for example, a "C" language, or a similar program design language. The program code can be completely executed on the computing device 200, partially executed on the computing device 200, executed as an independent software package, partially executed on the computing device 200 and partially executed on a remote computing device, or completely executed on a remote computing device.

[0142] Specific embodiments of this specification are described above. Other embodiments fall within the scope of the appended claims. In some cases, the actions or steps described in the

claims can be performed in a sequence different from that in the embodiments and desired results can still be achieved. In addition, the process depicted in the accompanying drawings does not necessarily need a particular sequence or consecutive sequence to achieve the desired results. In some implementations, multitasking and parallel processing are possible or may be advantageous.

5 **[0143]** In conclusion, after reading the detailed disclosure, a person skilled in the art can understand that the detailed disclosure can be presented only by using an example, and can impose no limitation. Although it is not explicitly stated herein, a person skilled in the art can understand that the requirements of this specification include various proper changes, improvements, and modifications to the embodiments. These changes, improvements, and modifications are intended
10 to be provided in this specification, and fall within the spirit and scope of the example embodiments of this specification.

[0144] In addition, some terms in this specification are used to describe the embodiments of this specification. For example, "one embodiment", "embodiments", and/or "some embodiments" mean/means that specific features, structures, or characteristics described with reference to the
15 embodiment can be included in at least one embodiment of this specification. Therefore, it can be emphasized and understood that two or more references to "embodiments", "one embodiment", or "alternative embodiments" in various parts of this specification do not necessarily refer to a same embodiment. In addition, specific features, structures, or characteristics can be appropriately combined in one or more embodiments of this specification.

20 **[0145]** It should be understood that in the above-mentioned descriptions of the embodiments of this specification, to help understand a feature, for the purpose of simplifying this specification, various features are combined in a single embodiment, accompanying drawing, or description thereof in this specification. However, this does not mean that the combination of these features is necessary. It is entirely possible for a person skilled in the art to mark some devices and understand
25 the devices as separate embodiments when reading this specification. That is, the embodiments of this specification can also be understood as an integration of a plurality of secondary embodiments. Content of each secondary embodiment is also true when a quantity of features is less than a quantity of all features in one of the embodiments disclosed above.

[0146] Each patent, patent application, publication of a patent application, and other materials,
30 for example, articles, books, instructions, publications, documents, and products, that are cited in this specification and that are other than any document that is inconsistent or conflict with this

document or that imposes limiting impact on the widest scope of the claims can be incorporated herein by reference and used for all purposes now or later associated with this document. In addition, the terms in this document shall prevail if there is any inconsistency or conflict between the descriptions, definitions, and/or use of the related terms in any material and the descriptions, definitions, and/or use of the related terms in this document.

[0147] Finally, it should be understood that the principles of the implementation solutions of this specification are described in the implementation solutions of the application disclosed in this specification. Other modified embodiments also fall within the scope of this specification. Therefore, the embodiments disclosed in this specification are merely examples and not limitations.

A person skilled in the art can use an alternative configuration according to the embodiments of this specification to implement the application in this specification. Therefore, the embodiments of this specification are not limited to the embodiments accurately described in the application.

CLAIMS

What is claimed is:

1. A traffic limiting method, applied to an intermediary platform, and comprising:
 - obtaining a target task of a user, wherein the target task is used to represent invocation of a
 - 5 target interface of a third-party platform to perform a target operation;
 - obtaining traffic limiting information of the target interface;
 - generating a current token for the target task based on the traffic limiting information; and
 - when the current token is a valid token that meets the traffic limiting information, invoking
 - the target interface to execute the target task.
- 10 2. The method according to claim 1, wherein the generating a current token for the target task based on the traffic limiting information comprises:
 - determining a current traffic window for the target task based on the traffic limiting
 - information; and
 - generating the current token based on historical token generation information in the current
 - 15 traffic window.
3. The method according to claim 2, wherein the historical token generation information comprises a quantity of historical tokens generated in the current traffic window; and the generating the current token based on historical token generation information in the current traffic window comprises:
 - 20 determining a current sequence number of the current token based on the quantity of historical
 - tokens; and
 - generating a token identifier of the current token based on the current sequence number, an
 - interface identifier of the target interface, and a current time, wherein the current token comprises
 - the token identifier.
- 25 4. The method according to claim 3, wherein the traffic limiting information is used to represent a maximum quantity of invocation times of the target interface in a preset time window; and the method further comprises:
 - if the maximum quantity of invocation times is greater than the current sequence number,
 - determining that the current token is a valid token.
- 30 5. The method according to claim 4, wherein the method further comprises:

if the quantity of invocation times is less than the current sequence number, adding the target task to a target asynchronous task list for re-execution.

6. The method according to claim 1, wherein the method further comprises:

obtaining a user instruction initiated by the user, wherein the user instruction is used to
5 indicate to perform a corresponding operation on the third-party platform;

querying an initial asynchronous task queue to determine whether the initial asynchronous task queue comprises a task corresponding to the user instruction, to obtain a query result; and

if the query result is that the initial asynchronous task queue comprises no task corresponding to the user instruction, adding the target task corresponding to the user instruction to the initial
10 asynchronous task list, to obtain a target asynchronous task list that comprises the target task.

7. The method according to claim 6, wherein the method further comprises:

if the query result is that the initial asynchronous task queue comprises a task corresponding to the user instruction, outputting a prompt message to the user to prompt the user to try again later.

8. The method according to claim 1, wherein the method further comprises:

15 updating a task status of the target task in response to an execution result of the target task.

9. The method according to claim 8, wherein the method further comprises:

providing feedback based on an updated task status in response to a polling request for the target task.

10. A traffic limiting system, comprising:

20 at least one storage medium that stores at least one instruction set for traffic limiting; and
at least one processor that is communicatively connected to the at least one storage medium, wherein when the traffic limiting system runs, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set:

25 obtaining a target task of a user, wherein the target task is used to represent invocation of a target interface of a third-party platform to perform a target operation;

obtaining traffic limiting information of the target interface;

generating a current token for the target task based on the traffic limiting information; and

when the current token is a valid token that meets the traffic limiting information, invoking
30 the target interface to execute the target task.

11. The system according to claim 10, wherein to generate the current token for the target

task based on the traffic limiting information, the at least one processor performs the following operations:

determining a current traffic window for the target task based on the traffic limiting information; and

5 generating the current token based on historical token generation information in the current traffic window.

12. The system according to claim 11, wherein to generate the current token based on the historical token generation information in the current traffic window, the at least one processor further performs the following operations:

10 determining a current sequence number of the current token based on a quantity of historical tokens; and

generating a token identifier of the current token based on the current sequence number, an interface identifier of the target interface, and a current time, wherein the current token comprises the token identifier.

15 13. The system according to claim 12, wherein the at least one processor further performs the following operation:

if a maximum quantity of invocation times is greater than the current sequence number, determining that the current token is a valid token.

20 14. The system according to claim 13, wherein the at least one processor further performs the following operation:

if the quantity of invocation times is less than the current sequence number, adding the target task to a target asynchronous task list for re-execution.

15. The system according to claim 10, wherein the at least one processor further performs the following operations:

25 obtaining a user instruction initiated by the user, wherein the user instruction is used to indicate to perform a corresponding operation on the third-party platform;

querying an initial asynchronous task queue to determine whether the initial asynchronous task queue comprises a task corresponding to the user instruction, to obtain a query result; and

30 if the query result is that the initial asynchronous task queue comprises no task corresponding to the user instruction, adding the target task corresponding to the user instruction to the initial asynchronous task list, to obtain a target asynchronous task list that comprises the target task.

16. The system according to claim 15, wherein the at least one processor further performs the following operation:
- if the query result is that the initial asynchronous task queue comprises a task corresponding to the user instruction, outputting a prompt message to the user to prompt the user to try again later.
- 5 17. The system according to claim 10, wherein the at least one processor further performs the following operation:
- updating a task status of the target task in response to an execution result of the target task.
18. The system according to claim 10, wherein the at least one processor further performs the following operation:
- 10 providing feedback based on an updated task status in response to a polling request for the target task.
19. A computer-readable nonvolatile storage medium, wherein the computer-readable nonvolatile storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the traffic limiting method according to any one of claims
- 15 1 to 9 is implemented.