

REQUEST SCHEDULING METHOD AND DEVICE

[0001] This application claims priority to Chinese Patent Application No. 202311386911.8, filed with the China National Intellectual Property Administration on October 24, 2023 and entitled "REQUEST SCHEDULING METHOD AND DEVICE", which is incorporated herein by
5 reference in its entirety.

TECHNICAL FIELD

[0002] The present invention relates to the field of computer technologies, and in particular, to a request scheduling method and a device.

BACKGROUND

[0003] A large language model (Large Language Model, LLM) has set off a new wave in the artificial intelligence (Artificial Intelligence, AI) field. An inference process of a generative pre-trained transformer (Generative Pre-trained Transformer, GPT) using a key-value cache (KV Cache) technology includes two phases: a first-round context filling (Prefill) phase and a multi-round decode (Decode) phase. In a large-scale language model service deployment, although a
10 service system uses a dynamic scheduling manner, a throughput of the service system in the LLM
15 service deployment is still low.

SUMMARY

[0004] In view of this, embodiments of the present invention provide a request scheduling method and a device, to improve a throughput of a service system in an LLM service deployment.

[0005] According to a first aspect, an embodiment of the present invention provides a request
20 scheduling method, applied to a request scheduling system, where the method includes:

[0006] A first device receives a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service;

the first device schedules the first request to a second device;

[0007] the second device sends a second request to the first device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request;
25

the first device schedules the second request to a third device when the first output information does not meet a specified condition, where the third device is different from the second device;

the third device sends a third request to the first device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request; and

[0007] The first device schedules the third request to the third device when the second output information does not meet the specified condition.

[0008] In addition, in this embodiment of the present invention, in the request scheduling system, the second device and the third device are deployed in an independent separate deployment manner. The first device schedules a request, so that the second device and the third device respectively perform inference processes in respective phases. In addition, information may be transferred between the two devices, so that the inference process can be normally performed, and a throughput of a service system can be improved.

[0009] With reference to the first aspect, in some implementations of the first aspect, the method further includes:

the first device receives a fourth request from a second user, where the fourth request is a request for a second service, and the fourth request includes an identifier of the second service;

the first device schedules the fourth request to the second device;

the second device sends a fifth request to the first device, where the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request;

the first device schedules the fifth request to the third device when the third output information does not meet the specified condition;

the third device sends a sixth request to the first device, where the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request; and

the first device schedules the sixth request to the third device when the fourth output information does not meet the specified condition.

[0010] In this embodiment of the present invention, the second user is different from the first user, and the first device can schedule requests from different users to the second device and the third device for processing, without waiting for the second device and/or the third device to process a request from one user before processing a request from another user. Therefore, a throughput of the service system can be improved.

[0011] With reference to the first aspect, in some implementations of the first aspect, the

method further includes:

the first device sends the second output information to the first user when the second output information meets the specified condition.

[0012] In this embodiment of the present invention, the first device determines whether the output information of the service meets the specified condition. If the specified condition is not met, the first device continues to perform scheduling and inference computing on the service. If the specified condition is met, the first device performs early stopping on the service, and sends the second output information to the first user. After early stopping the service, the first device may indicate the third device to release a resource corresponding to the service. The third device reports a current resource status of the third device to the first device. If there is an idle computing service resource in the current resource status, the first device continues to schedule a service to the third device to insert the idle computing service resource, to implement dynamic insertion.

[0013] With reference to the first aspect, in some implementations of the first aspect, the request scheduling system includes the first device, at least one second device, and at least one third device. In this embodiment of the present invention, when there are a large quantity of second devices and third devices in the request scheduling system, the second devices and the third devices are deployed in an independent separate deployment manner. A small batch or a single batch is configured for the second devices, and a large batch is configured for the third devices. A resource proportion of the second device resource to the third device resource may be dynamically adjusted based on system load, to obtain a maximum throughput, and greatly improve the throughput.

[0014] According to a second aspect, an embodiment of the present invention provides a request scheduling method, used in a first device, where the method includes:

receiving a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service;

scheduling the first request to a second device;

receiving a second request sent by the second device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request;

scheduling the second request to a third device when the first output information does not meet a specified condition, where the third device is different from the second device;

receiving a third request sent by the third device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request; and

scheduling the third request to the third device when the second output information does not meet the specified condition.

[0015] In this embodiment of the present invention, a request for a new service from a user is scheduled to the second device, that is, a prefill inference execution container performs inference computing in a prefill phase; and a request for an old service that passes through the prefill phase is scheduled to the third device, that is, a decode inference execution container performs inference computing in a decoding phase. In the present invention, in the separate deployment manner, a significant difference between a computing power bottleneck in the prefill phase and a memory access bottleneck in the decoding phase is comprehensively considered, so that the throughput of the service system can be improved.

[0016] With reference to the second aspect, in some implementations of the second aspect, the method further includes:

receiving a fourth request from a second user, where the fourth request is a request for a second service, and the fourth request includes an identifier of the second service;

scheduling the fourth request to the second device;

receiving a fifth request sent by the second device, where the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request;

scheduling the fifth request to the third device when the third output information does not meet the specified condition;

receiving a sixth request sent by the third device, where the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request; and

scheduling the sixth request to the third device when the fourth output information does not meet the specified condition.

[0017] In this embodiment of the present invention, the second user is different from the first user, and the first device can schedule requests from different users to the second device and the third device for processing, without waiting for the second device and/or the third device to process a request from one user before processing a request from another user. Therefore, a throughput of the service system can be improved.

[0018] With reference to the second aspect, in some implementations of the second aspect, the method further includes:

when the second output information meets the specified condition, sending the second output information to the first user.

[0019] In this embodiment of the present invention, the first device determines whether the output information of the service meets the specified condition. If the specified condition is not met, the first device continues to perform scheduling and inference computing on the service. If

the specified condition is met, the first device performs early stopping on the service, and sends the second output information to the first user. After early stopping the service, the first device may indicate the third device to release a resource corresponding to the service. The third device reports a current resource status of the third device to the first device. If there is an idle computing service resource in the current resource status, the first device continues to schedule a service to the third device to insert the idle computing service resource, to implement dynamic insertion.

[0020] With reference to the second aspect, in some implementations of the second aspect, the identifier includes a service ID.

[0021] With reference to the second aspect, in some implementations of the second aspect, the specified condition includes at least one of the following:

a length of output information is greater than or equal to a first threshold; or
the output information includes a terminator.

[0022] According to a third aspect, an embodiment of the present invention provides a request scheduling method, used in a second device, where the method includes:

receiving a first request sent by a first device from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service; and
sending a second request to the first device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request.

[0023] In this embodiment of the present invention, a request for a new service from a user is scheduled to the second device, that is, a prefill inference execution container performs inference computing in a prefill phase; and a request for an old service that passes through the prefill phase is scheduled to the third device, that is, a decode inference execution container performs inference computing in a decoding phase. In the present invention, in the separate deployment manner, a significant difference between a computing power bottleneck in the prefill phase and a memory access bottleneck in the decoding phase is comprehensively considered, so that the throughput of the service system can be improved.

[0024] With reference to the third aspect, in some implementations of the third aspect, the method further includes:

receiving a fourth request sent by the first device from a second user, where the fourth request is a request for a second service, and the fourth request includes an identifier of the second service; and

sending a fifth request to the first device, where the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request.

[0025] In this embodiment of the present invention, the second user is different from the first user, and the first device can schedule requests from different users to the second device and the third device for processing, without waiting for the second device and/or the third device to process a request from one user before processing a request from another user. Therefore, a throughput of the service system can be improved.

[0026] With reference to the third aspect, in some implementations of the third aspect, before the sending a second request to the first device, the method further includes:

performing, based on a first computing resource status and a first condition, group batching on the first request and at least one seventh request that arrives at the second device at the same time as the first request, to obtain a first batch, where the seventh request is a request from the first user, a second user, or another user; and

performing first inference computing on the first batch to obtain the second request and first cache information.

[0027] With reference to the third aspect, in some implementations of the third aspect, the method further includes:

sending the first cache information to the third device.

[0028] With reference to the third aspect, in some implementations of the third aspect, the cache information includes key-value KV cache information.

[0029] With reference to the third aspect, in some implementations of the third aspect, the performing, based on a first computing resource status and a first condition, group batching on the first request and at least one seventh request that arrives at the second device at the same time as the first request, to obtain a first batch includes:

obtaining, based on the first request, input information corresponding to the first request and a first length of the input information; and

performing group batching on the first request and the at least one seventh request based on the first computing resource status, the first condition, and the first length, to obtain the first batch. A length of the input information corresponding to the seventh request is similar to the first length. This helps reduce a waste of idle resources in a first inference computing process.

[0030] With reference to the third aspect, in some implementations of the third aspect, the first computing resource status includes a quantity of idle resources in the second device.

[0031] With reference to the third aspect, in some implementations of the third aspect, the first condition includes a maximum quantity of requests grouped by the second device in a batch.

[0032] With reference to the third aspect, in some implementations of the third aspect, in the first processing manner, after inference computing of all requests in a current batch ends, computing service resources occupied by the current batch are released, and a next batch is inserted

into the idle computing service resources to perform first inference computing.

[0033] In this embodiment of the present invention, for inference computing in the prefill phase, inference computing is performed based on a single batch. To be specific, after inference computing of all requests in a current batch ends, computing service resources occupied by the current batch are released, and a next batch is inserted into an idle computing service resource for starting to perform the first inference computing on the next batch.

[0034] According to a fourth aspect, an embodiment of the present invention provides a request scheduling method, used in a third device, where the method includes:

receiving a second request sent by a first device from a second device, where the second request includes an identifier of a first service and first output information corresponding to the first service, the second request is associated with the first request, and the first request is a request for the first service from a first user; and

sending a third request to the first device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request.

[0035] In this embodiment of the present invention, a request for a new service from a user is scheduled to the second device, that is, a prefill inference execution container performs inference computing in a prefill phase; and a request for an old service that passes through the prefill phase is scheduled to the third device, that is, a decode inference execution container performs inference computing in a decoding phase. In the present invention, in the separate deployment manner, a significant difference between a computing power bottleneck in the prefill phase and a memory access bottleneck in the decoding phase is comprehensively considered, so that the throughput of the service system can be improved.

[0036] With reference to the fourth aspect, in some implementations of the fourth aspect, the method further includes:

receiving a fifth request sent by the first device, where the fifth request includes an identifier of a second service and third output information corresponding to the second service, the fifth request is associated with a fourth request, and the fourth request is a request for the second service from a second user; and

sending a sixth request to the first device, where the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request.

[0037] In this embodiment of the present invention, the second user is different from the first user, and the first device can schedule requests from different users to the second device and the third device for processing, without waiting for the second device and/or the third device to process

a request from one user before processing a request from another user. Therefore, a throughput of the service system can be improved.

[0038] With reference to the fourth aspect, in some implementations of the fourth aspect, before the sending a third request to the first device, the method further includes:

5 performing, based on a second computing resource status and a second condition, group batching on the second request and at least one eighth request that arrives at the third device at the same time as the second request, to obtain a second batch, where the eighth request is from the second device or the third device;

receiving first cache information sent by the second device; and

10 performing second inference computing on the second batch based on the first cache information, to obtain the third request and second cache information.

[0039] According to a fifth aspect, an embodiment of the present invention provides a request scheduling system, where the system includes:

15 a first device, configured to receive a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service;

the first device, further configured to schedule the first request to a second device;

the second device, configured to send a second request to the first device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request;

20 the first device, further configured to schedule the second request to a third device when the first output information does not meet a specified condition, where the third device is different from the second device;

the third device, configured to send a third request to the first device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request; and

25 the first device is further configured to schedule the third request to the third device when the second output information does not meet the specified condition.

[0040] According to a sixth aspect, an embodiment of the present invention provides a device, including a processor and a memory, where the memory is configured to store a program, the program includes program instructions, and when the processor runs the program instructions, the device is enabled to perform the steps of the foregoing method.

30 **[0041]** According to a seventh aspect, an embodiment of the present invention provides a readable storage medium, where the readable storage medium stores a program, the program includes program instructions, and when the program requests are run by a device, the device is enabled to perform the foregoing method.

[0042] According to an eighth aspect, an embodiment of the present invention provides a program product, where the program product includes instructions, and when the program product runs on a device or at least one processor, the device is enabled to perform the function/step in the foregoing method.

5 **[0043]** In the technical solutions of the request scheduling method and the device provided in embodiments of the present invention. A first device receives a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service; the first device schedules the first request to a second device; the second device sends a second request to the first device, where the second request includes the identifier of the first
10 service and first output information, and the second request is associated with the first request; when the first output information does not meet a specified condition, the first device schedules the second request to a third device, where the third device is different from the second device; the third device sends a third request to the first device, where the third request includes the identifier of the first service and second output information, and the third request is associated with the
15 second request; and when the second output information does not meet the specified condition, the first device schedules the third request to the third device, so that a throughput of a service system in a large-scale language model service deployment can be improved.

BRIEF DESCRIPTION OF DRAWINGS

[0044] FIG. 1 is a diagram of a structure of a device according to an embodiment of the present
20 invention;

[0045] FIG. 2 is a block diagram of a software structure of a device 100 according to an embodiment of the present invention;

[0046] FIG. 3 is a diagram of an architecture of a request scheduling system according to an embodiment of the present invention;

25 **[0047]** FIG. 4A and FIG. 4B are a signaling interaction diagram of a request scheduling method according to an embodiment of the present invention;

[0048] FIG. 5 is a diagram of prefill inference and decode inference according to an embodiment of the present invention;

[0049] FIG. 6A and FIG. 6B are a signaling interaction diagram of another request scheduling
30 method according to an embodiment of the present invention;

[0050] FIG. 7A and FIG. 7B are a flowchart of a request scheduling method according to an embodiment of the present invention;

[0051] FIG. 8 is a flowchart of another request scheduling method according to an embodiment

of the present invention;

[0052] FIG. 9 is a diagram of a structure of a first device according to an embodiment of the present invention;

[0053] FIG. 10 is a diagram of a structure of a second device according to an embodiment of the present invention; and

[0054] FIG. 11 is a diagram of a structure of a third device according to an embodiment of the present invention.

DESCRIPTION OF EMBODIMENTS

[0055] To make the technical solutions in the present invention more comprehensible, the following describes embodiments of the present invention in detail with reference to the accompanying drawings.

[0056] It should be clear that the described embodiments are merely some but not all of embodiments of the present invention. All other embodiments obtained by persons of ordinary skill in the art based on the embodiments of the present invention without creative efforts shall fall within the protection scope of the present invention.

[0057] The terms used in embodiments of the present invention are merely for the purpose of illustrating specific embodiments, and are not intended to limit the present invention. The terms "a", "said" and "the" of singular forms used in embodiments and the appended claims of the present invention are also intended to include plural forms, unless otherwise specified in the context clearly.

[0058] It should be understood that the term "and/or" used in this specification describes only an association relationship between associated objects and represents that three relationships may exist. For example, A and/or B may represent the following three cases: Only A exists, both A and B exist, and only B exists. In addition, the character "/" in this specification generally indicates an "or" relationship between the associated objects.

[0059] FIG. 1 is a diagram of a structure of a device 100.

[0060] The device 100 may include a processor 110, an external memory interface 120, an internal memory 121, a universal serial bus (universal serial bus, USB) interface 130, a charging management module 140, a power management module 141, a battery 142, an antenna 1, an antenna 2, a mobile communication module 150, a wireless communication module 160, an audio module 170, a speaker 170A, a receiver 170B, a microphone 170C, a headset jack 170D, a sensor module 180, a button 190, a motor 191, an indicator 192, a camera 193, a display 194, a subscriber identity module (subscriber identity module, SIM) card interface 195, and the like. The sensor module 180 may include a pressure sensor 180A, a gyro sensor 180B, a barometric pressure sensor

180C, a magnetic sensor 180D, an acceleration sensor 180E, a distance sensor 180F, an optical proximity sensor 180G, a fingerprint sensor 180H, a temperature sensor 180J, a touch sensor 180K, an ambient light sensor 180L, a bone conduction sensor 180M, and the like.

[0061] It can be understood that the structure shown in this embodiment of the present invention does not constitute a specific limitation on the device 100. In some other embodiments of this application, the device 100 may include more or fewer components than those shown in the figure, or combine some components, or split some components, or have different component arrangements. The components shown in the figure may be implemented by hardware, software, or a combination of software and hardware.

[0062] The processor 110 may include one or more processing units. For example, the processor 110 may include an application processor (application processor, AP), a modem processor, a graphics processing unit (graphics processing unit, GPU), an image signal processor (image signal processor, ISP), a controller, a video codec, a digital signal processor (digital signal processor, DSP), a baseband processor, a neural-network processing unit (neural-network processing unit, NPU), and/or the like. Different processing units may be independent components, or may be integrated into one or more processors.

[0063] The controller may generate an operation control signal based on an instruction operation code and a time sequence signal, to complete control of instruction reading and instruction execution.

[0064] A memory may be further disposed in the processor 110, and is configured to store instructions and data. In some embodiments, the memory in the processor 110 is a cache memory. The memory may store instructions or data that have/has been used or cyclically used by the processor 110. If the processor 110 needs to use the instructions or the data again, the processor may directly invoke the instructions or the data from the memory. This avoids repeated access, reduces waiting time of the processor 110, and improves system efficiency.

[0065] In some embodiments, the processor 110 may include one or more interfaces. The interface may include an inter-integrated circuit (inter-integrated circuit, I2C) interface, an inter-integrated circuit sound (inter-integrated circuit sound, I2S) interface, a pulse code modulation (pulse code modulation, PCM) interface, a universal asynchronous receiver/transmitter (universal asynchronous receiver/transmitter, UART) interface, a mobile industry processor interface (mobile industry processor interface, MIPI), a general-purpose input/output (general-purpose input/output, GPIO) interface, a subscriber identity module (subscriber identity module, SIM) interface, a universal serial bus (universal serial bus, USB) interface, and/or the like.

[0066] The I2C interface is a bidirectional synchronous serial bus, including a serial data line (serial data line, SDA) and a serial clock line (serial clock line, SCL). In some embodiments, the

processor 110 may include a plurality of groups of I2C buses. The processor 110 may be separately coupled to the touch sensor 180K, a charger, a flash, the camera 193, and the like through different I2C bus interfaces. For example, the processor 110 may be coupled to the touch sensor 180K through the I2C interface, so that the processor 110 communicates with the touch sensor 180K through an I2C bus interface, to implement a touch function of the device 100.

[0067] The I2S interface may be configured to perform audio communication. In some embodiments, the processor 110 may include a plurality of groups of I2S buses. The processor 110 may be coupled to the audio module 170 through the I2S bus, to implement communication between the processor 110 and the audio module 170. In some embodiments, the audio module 170 may transmit a sound signal to the wireless communication module 160 through the I2S interface, to implement a function of answering a call through a Bluetooth headset.

[0068] The PCM interface may also be used to perform audio communication, and sample, quantize, and code an analog signal. In some embodiments, the audio module 170 may be coupled to the wireless communication module 160 through a PCM bus interface. In some embodiments, the audio module 170 may also transmit a sound signal to the wireless communication module 160 through the PCM interface, to implement a function of answering a call through a Bluetooth headset. Both the I2S interface and the PCM interface may be used for audio communication.

[0069] The UART interface is a universal serial data bus, and is configured to perform asynchronous communication. The bus may be a two-way communication bus. The bus converts to-be-transmitted data between serial communication and parallel communication. In some embodiments, the UART interface is usually configured to connect the processor 110 to the wireless communication module 160. For example, the processor 110 communicates with a Bluetooth module in the wireless communication module 160 through the UART interface, to implement a Bluetooth function. In some embodiments, the audio module 170 may transmit a sound signal to the wireless communication module 160 through the UART interface, to implement a function of playing music through a Bluetooth headset.

[0070] The MIPI interface may be configured to connect the processor 110 to a peripheral component such as the display 194 or the camera 193. The MIPI interface includes a camera serial interface (camera serial interface, CSI), a display serial interface (display serial interface, DSI), and the like. In some embodiments, the processor 110 communicates with the camera 193 through the CSI interface, to implement a photographing function of the device 100. The processor 110 communicates with the display 194 through the DSI interface, to implement a display function of the device 100.

[0071] The GPIO interface may be configured by software. The GPIO interface may be configured as a control signal or a data signal. In some embodiments, the GPIO interface may be

configured to connect the processor 110 to the camera 193, the display 194, the wireless communication module 160, the audio module 170, the sensor module 180, or the like. The GPIO interface may alternatively be configured as an I2C interface, an I2S interface, a UART interface, an MIPI interface, or the like.

5 **[0072]** The USB interface 130 is an interface that conforms to a USB standard specification, and may be specifically a mini USB interface, a micro USB interface, a USB type-C interface, or the like. The USB interface 130 may be used to be connected to the charger to charge the device 100, or may be used to transmit data between the device 100 and a peripheral device, or may be configured to connect to a headset for playing audio through the headset. The interface may be
10 further used to connect to another device, like an AR device.

[0073] It may be understood that an interface connection relationship between the modules that is shown in this embodiment of the present invention is merely an example for description, and does not constitute a limitation on a structure of the device 100. In some other embodiments of this application, the device 100 may alternatively use an interface connection manner different
15 from that in the foregoing embodiment, or use a combination of a plurality of interface connection manners.

[0074] The charging management module 140 is configured to receive a charging input from a charger. The charger may be a wireless charger or a wired charger. In some embodiments of wired charging, the charging management module 140 may receive a charging input of a wired
20 charger through the USB interface 130. In some embodiments of wireless charging, the charging management module 140 may receive a wireless charging input by using a wireless charging coil of the device 100. The charging management module 140 may further supply power to the device by using the power management module 141 while charging the battery 142.

[0075] The power management module 141 is configured to connect to the battery 142, the
25 charging management module 140, and the processor 110. The power management module 141 receives an input from the battery 142 and/or the charging management module 140, and supplies power to the processor 110, the internal memory 121, the display 194, the camera 193, the wireless communication module 160, and the like. The power management module 141 may be further configured to monitor parameters such as a battery capacity, a battery cycle count, and a battery
30 health status (electric leakage or impedance). In some other embodiments, the power management module 141 may alternatively be disposed in the processor 110. In some other embodiments, the power management module 141 and the charging management module 140 may alternatively be disposed in a same device.

[0076] A wireless communication function of the device 100 may be implemented through the
35 antenna 1, the antenna 2, the mobile communication module 150, the wireless communication

module 160, the modem processor, the baseband processor, and the like.

[0077] The antenna 1 and the antenna 2 are configured to transmit and receive an electromagnetic wave signal. Each antenna in the device 100 may be configured to cover one or more communication frequency bands. Different antennas may be further multiplexed, to improve antenna utilization. For example, the antenna 1 may be multiplexed as a diversity antenna of a wireless local area network. In some other embodiments, the antenna may be used in combination with a tuning switch.

[0078] The mobile communication module 150 may provide a wireless communication solution that is used in the device 100, including 2G/3G/4G/5G and the like. The mobile communication module 150 may include at least one filter, a switch, a power amplifier, a low noise amplifier (low noise amplifier, LNA), and the like. The mobile communication module 150 may receive an electromagnetic wave through the antenna 1, perform processing such as filtering or amplification on the received electromagnetic wave, and transmit the electromagnetic wave to the modem processor for demodulation. The mobile communication module 150 may further amplify a signal modulated by the modem processor, and convert the signal into an electromagnetic wave for radiation through the antenna 1. In some embodiments, at least some functional modules in the mobile communication module 150 may be disposed in the processor 110. In some embodiments, at least some functional modules of the mobile communication module 150 may be disposed in a same device as at least some modules of the processor 110.

[0079] The modem processor may include a modulator and a demodulator. The modulator is configured to modulate a to-be-sent low-frequency baseband signal into a medium-high frequency signal. The demodulator is configured to demodulate a received electromagnetic wave signal into a low-frequency baseband signal. Then, the demodulator transmits the low-frequency baseband signal obtained through demodulation to the baseband processor for processing. The low-frequency baseband signal is processed by the baseband processor and then transmitted to the application processor. The application processor outputs a sound signal by an audio device (which is not limited to the speaker 170A, the receiver 170B, or the like), or displays an image or a video by the display 194. In some embodiments, the modem processor may be an independent component. In some other embodiments, the modem processor may be independent of the processor 110, and is disposed in a same device as the mobile communication module 150 or another functional module.

[0080] The wireless communication module 160 may provide a wireless communication solution that is used in the device 100, and that includes a wireless local area network (wireless local area networks, WLAN) (for example, a wireless fidelity (wireless fidelity, Wi-Fi) network), Bluetooth (Bluetooth, BT), a global navigation satellite system (global navigation satellite system,

GNSS), frequency modulation (frequency modulation, FM), a near field communication (near field communication, NFC) technology, an infrared (infrared, IR) technology, or the like. The wireless communication module 160 may be one or more components integrating at least one communication processor module. The wireless communication module 160 receives an electromagnetic wave by the antenna 2, performs frequency modulation and filtering processing on an electromagnetic wave signal, and sends a processed signal to the processor 110. The wireless communication module 160 may further receive a to-be-sent signal from the processor 110, perform frequency modulation and amplification on the signal, and convert the signal into an electromagnetic wave for radiation through the antenna 2.

[0081] In some embodiments, the antenna 1 and the mobile communication module 150 in the device 100 are coupled, and the antenna 2 and the wireless communication module 160 in the device 100 are coupled, so that the device 100 can communicate with a network and another device by using a wireless communication technology. The wireless communication technology may include a global system for mobile communications (global system for mobile communications, GSM), a general packet radio service (general packet radio service, GPRS), code division multiple access (code division multiple access, CDMA), wideband code division multiple access (wideband code division multiple access, WCDMA), time-division code division multiple access (time-division code division multiple access, TD-SCDMA), long term evolution (long term evolution, LTE), BT, a GNSS, a WLAN, NFC, FM, an IR technology, and/or the like. The GNSS may include a global positioning system (global positioning system, GPS), a global navigation satellite system (global navigation satellite system, GLONASS), a BeiDou navigation satellite system (BeiDou navigation satellite system, BDS), a quasi-zenith satellite system (quasi-zenith satellite system, QZSS), and/or a satellite based augmentation system (satellite based augmentation systems, SBAS).

[0082] The device 100 implements a display function by using the GPU, the display 194, the application processor, and the like. The GPU is a microprocessor for image processing, and is connected to the display 194 and the application processor. The GPU is configured to: perform mathematical and geometric computation, and render an image. The processor 110 may include one or more GPUs, which execute program instructions to generate or change display information.

[0083] The display 194 is configured to display an image, a video, and the like. The display 194 includes a display panel. The display panel may be a liquid crystal display (liquid crystal display, LCD), an organic light-emitting diode (organic light-emitting diode, OLED), an active-matrix organic light emitting diode (active-matrix organic light emitting diode, AMOLED), a flexible light-emitting diode (flexible light-emitting diode, FLED), a mini-LED, a micro-LED, a micro-OLED, a quantum dot light emitting diode (quantum dot light emitting diode, QLED), or

the like. In some embodiments, the device 100 may include one or N displays 194. N is a positive integer greater than 1.

[0084] The device 100 may implement a photographing function by using the ISP, the camera 193, the video codec, the GPU, the display 194, the application processor, and the like.

5 **[0085]** The ISP is configured to process data fed back by the camera 193. For example, during photographing, a shutter is pressed, and light is transmitted to a photosensitive element of the camera through a lens. An optical signal is converted into an electrical signal, and the photosensitive element of the camera transmits the electrical signal to the ISP for processing, to convert the electrical signal into a visible image. The ISP may further perform algorithm
10 optimization on noise, brightness, and complexion of the image. The ISP may further optimize parameters such as exposure and a color temperature of a photographing scenario. In some embodiments, the ISP may be disposed in the camera 193.

[0086] The camera 193 is configured to capture a static image or a video. An optical image of an object is generated through the lens, and is projected onto the photosensitive element. The
15 photosensitive element may be a charge coupled device (charge coupled device, CCD) or a complementary metal-oxide-semiconductor (complementary metal-oxide-semiconductor, CMOS) phototransistor. The light-sensitive element converts an optical signal into an electrical signal, and then transmits the electrical signal to the ISP to convert the electrical signal into a digital image signal. The ISP outputs the digital image signal to the DSP for processing. The DSP converts the
20 digital image signal into an image signal in a standard format such as RGB or YUV. In some embodiments, the device 100 may include one or N cameras 193. N is a positive integer greater than 1.

[0087] The digital signal processor is configured to process a digital signal, and may process another digital signal in addition to the digital image signal. For example, when the device 100
25 selects a frequency bin, the digital signal processor is configured to perform Fourier transformation or the like on energy at the frequency bin.

[0088] The video codec is configured to compress or decompress a digital video. The device 100 may support one or more video codecs. In this way, the device 100 may play or record videos in a plurality of coding formats, for example, moving picture experts group (moving picture
30 experts group, MPEG)-1, MPEG-2, MPEG-3, and MPEG-4.

[0089] The NPU is a neural-network (neural-network, NN) computing processor, quickly processes input information by referring to a structure of a biological neural network, for example, a transfer mode between human brain neurons, and may further continuously perform self-learning. Applications such as intelligent cognition of the device 100 may be implemented through the NPU,
35 for example, image recognition, facial recognition, speech recognition, and text understanding

[0090] The external memory interface 120 may be used to connect to an external storage card, for example, a micro SD card, to extend a storage capability of the device 100. The external memory card communicates with the processor 110 through the external memory interface 120, to implement a data storage function. For example, files such as music and videos are stored in the external storage card.

[0091] The internal memory 121 may be configured to store computer-executable program code. The executable program code includes instructions. The internal memory 121 may include a program storage area and a data storage area. The program storage area may store an operating system, an application required by at least one function (for example, a voice playing function or an image playing function), and the like. The data storage area may store data (such as audio data and an address book) created during use of the device 100, and the like. In addition, the internal memory 121 may include a high-speed random access memory, or may include a nonvolatile memory, for example, at least one magnetic disk storage device, a flash memory, or a universal flash storage (universal flash storage, UFS). The processor 110 runs instructions stored in the internal memory 121 and/or instructions stored in the memory disposed in the processor, to perform various function applications and data processing of the device 100.

[0092] The device 100 may implement an audio function, for example, music playing and recording, through the audio module 170, the speaker 170A, the receiver 170B, the microphone 170C, the headset jack 170D, the application processor, and the like.

[0093] The audio module 170 is configured to convert digital audio information into an analog audio signal for output, and is also configured to convert analog audio input into a digital audio signal. The audio module 170 may be further configured to code and decode an audio signal. In some embodiments, the audio module 170 may be disposed in the processor 110, or some functional modules in the audio module 170 are disposed in the processor 110.

[0094] The speaker 170A, also referred to as a "loudspeaker", is configured to convert an audio electrical signal into a sound signal. The device 100 may be used to listen to music or answer a call in a hands-free mode over the speaker 170A.

[0095] The receiver 170B, also referred to as an "earpiece", is configured to convert an electrical audio signal into a sound signal. When a call is answered or a voice message is received through the device 100, the receiver 170B may be put close to a human ear to listen to a voice.

[0096] The microphone 170C, also referred to as a "mike" or a "mic", is configured to convert a sound signal into an electrical signal. When making a call or sending a voice message, a user may make a sound near the microphone 170C through the mouth of the user, to input a sound signal to the microphone 170C. At least one microphone 170C may be disposed on the device 100.

In some other embodiments, two microphones 170C may be disposed in the device 100, to collect

a sound signal and further implement a noise reduction function. In some other embodiments, three, four, or more microphones 170C may alternatively be disposed in the device 100, to collect a sound signal, reduce noise, identify a sound source, implement a directional recording function, and the like.

5 **[0097]** The headset jack 170D is configured to connect to a wired headset. The headset jack 170D may be a USB interface 130, or may be a 3.5 mm open mobile terminal platform (open mobile terminal platform, OMTF) standard interface or cellular telecommunications industry association of the USA (cellular telecommunications industry association of the USA, CTIA) standard interface.

10 **[0098]** The pressure sensor 180A is configured to sense a pressure signal, and can convert the pressure signal into an electrical signal. In some embodiments, the pressure sensor 180A may be disposed on the display 194. There are a plurality of types of pressure sensors 180A, such as a resistive pressure sensor, an inductive pressure sensor, and a capacitive pressure sensor. The capacitive pressure sensor may include at least two parallel plates made of conductive materials.

15 When a force is applied to the pressure sensor 180A, capacitance between electrodes changes. The device 100 determines pressure intensity based on a capacitance change. When a touch operation is performed on the display 194, the device 100 detects intensity of the touch operation through the pressure sensor 180A. The device 100 may calculate a touch position based on a detection signal of the pressure sensor 180A. In some embodiments, touch operations that are performed in

20 a same touch position but have different touch operation intensity may correspond to different operation instructions. For example, when a touch operation whose touch operation intensity is less than a first pressure threshold is performed on an SMS message application icon, an instruction for viewing an SMS message is performed. When a touch operation whose touch operation intensity is greater than or equal to the first pressure threshold is performed on the SMS message

25 application icon, an instruction for creating a new SMS message is performed.

[0099] The gyro sensor 180B may be configured to determine a motion posture of the device 100. In some embodiments, angular velocities of the device 100 around three axes (which are x, y, and z axes) may be determined by using the gyro sensor 180B. The gyro sensor 180B may be configured to implement image stabilization during photographing. For example, when the shutter

30 is pressed, the gyro sensor 180B detects a shake angle of the device 100, and calculates, based on the angle, a distance for which a lens module needs to compensate, so that the lens counteracts the shake of the device 100 through reverse motion, implementing image stabilization. The gyro sensor 180B may also be used in a navigation scenario and a somatic game scenario.

[00100] The barometric pressure sensor 180C is configured to measure barometric pressure. In

35 some embodiments, the device 100 calculates an altitude based on a value of the barometric

pressure measured by the barometric pressure sensor 180C, to assist in positioning and navigation.

[00101] The magnetic sensor 180D includes a Hall sensor. The device 100 may detect opening and closing of a flip cover by using the magnetic sensor 180D. In some embodiments, when the device 100 is a flip device, the device 100 may detect opening and closing of flip by using the magnetic sensor 180D. Further, a feature such as automatic unlocking of the flip cover is set based on a detected opening or closing state of the leather case or a detected opening or closing state of the flip cover.

[00102] The acceleration sensor 180E may detect magnitudes of accelerations of the device 100 in all directions (usually on three axes). When the device 100 is still, a magnitude and a direction of gravity may be detected. The acceleration sensor 180E may be further configured to recognize a posture of the device, and is used in screen switching between a landscape mode and a portrait mode, a pedometer, or another application.

[00103] The distance sensor 180F is configured to measure a distance. The device 100 may measure a distance through infrared or laser. In some embodiments, in a photographing scenario, the device 100 may measure the distance by using the distance sensor 180F, to implement fast focusing.

[00104] The optical proximity sensor 180G may include, for example, a light emitting diode (LED) and an optical detector, for example, a photodiode. The light emitting diode may be an infrared light emitting diode. The device 100 emits infrared light outward by using the light-emitting diode. The device 100 detects infrared reflected light from a nearby object by using the photodiode. When sufficient reflected light is detected, the device 100 may determine that there is an object near the device 100. When insufficient reflected light is detected, the device 100 may determine that there is no object near the device 100. The device 100 may use the optical proximity sensor 180G to detect that a user holds the device 100 close to an ear for a call, to automatically turn off the display screen to save power. The optical proximity sensor 180G may also be used in a smart cover mode or a pocket mode to automatically perform screen unlocking or locking.

[00105] The ambient light sensor 180L is configured to sense ambient light brightness. The device 100 may adaptively adjust brightness of the display 194 based on the sensed ambient light brightness. The ambient light sensor 180L may also be configured to automatically adjust white balance during photographing. The ambient light sensor 180L may further cooperate with the optical proximity sensor 180G to detect whether the device 100 is in a pocket, to avoid an unintentional touch.

[00106] The fingerprint sensor 180H is configured to collect a fingerprint. The device 100 may use a feature of the collected fingerprint to implement fingerprint-based unlocking, application lock access, fingerprint-based photographing, fingerprint-based call answering, and the like.

[00107] The temperature sensor 180J is configured to detect a temperature. In some embodiments, the device 100 executes a temperature processing policy by using the temperature detected by the temperature sensor 180J. For example, when the temperature reported by the temperature sensor 180J exceeds a threshold, the device 100 degrades performance of a processor near the temperature sensor 180J, to reduce power consumption for thermal protection. In some other embodiments, when the temperature is less than another threshold, the device 100 heats the battery 142 to avoid abnormal shutdown of the device 100 caused by a low temperature. In some other embodiments, when the temperature is less than still another threshold, the device 100 boosts an output voltage of the battery 142 to avoid abnormal shutdown caused by a low temperature.

[00108] The touch sensor 180K is also referred to as a "touch device". The touch sensor 180K may be disposed on the display 194, and the touch sensor 180K and the display 194 constitute a touchscreen, which is also referred to as a "touch screen". The touch sensor 180K is configured to detect a touch operation performed on or near the touch sensor. The touch sensor may transfer the detected touch operation to the application processor to determine a type of the touch event. A visual output related to the touch operation may be provided through the display 194. In some other embodiments, the touch sensor 180K may alternatively be disposed on a surface of the device 100 at a position different from a position of the display 194.

[00109] The bone conduction sensor 180M may obtain a vibration signal. In some embodiments, the bone conduction sensor 180M may obtain a vibration signal of a vibration bone of a human vocal-cord part. The bone conduction sensor 180M may also be in contact with a body pulse to receive a blood pressure beating signal. In some embodiments, the bone conduction sensor 180M may also be disposed in the headset, to obtain a bone conduction headset. The audio module 170 may obtain a speech signal through parsing based on the vibration signal that is of the vibration bone of the vocal-cord part and that is obtained by the bone conduction sensor 180M, to implement a speech function. The application processor may parse heart rate information based on the blood pressure beating signal obtained by the bone conduction sensor 180M, to implement a heart rate detection function.

[00110] The button 190 includes a power button, a volume button, and the like. The button 190 may be a mechanical button, or may be a touch button. The device 100 may receive a button input, and generate a button signal input related to user settings and function control of the device 100.

[00111] The motor 191 may generate a vibration prompt. The motor 191 may be configured to provide an incoming call vibration prompt and a touch vibration feedback. For example, touch operations performed on different applications (for example, photographing and audio playing) may correspond to different vibration feedback effects. The motor 191 may also correspond to different vibration feedback effects for touch operations performed on different areas of the display

194. Different application scenarios (for example, a time reminder, information receiving, an alarm clock, and a game) may also correspond to different vibration feedback effects. A touch vibration feedback effect may be further customized.

[00112] The indicator 192 may be an indicator light, and may be configured to indicate a charging status and a power change, or may be configured to indicate a message, a missed call, a notification, and the like.

[00113] The SIM card interface 195 is configured to connect to a SIM card. The SIM card may be inserted into the SIM card interface 195 or pulled out of the SIM card interface 195, so that the SIM card is in contact with and separated from the device 100. The device 100 may support one or N SIM card interfaces. N is a positive integer greater than 1. The SIM card interface 195 may support a nano-SIM card, a micro-SIM card, a SIM card, and the like. A plurality of cards may be inserted into a same SIM card interface 195 at the same time. The plurality of cards may be of a same type or different types. The SIM card interface 195 may be compatible with different types of SIM cards. The SIM card interface 195 is also compatible with an external storage card. The device 100 interacts with the network by using the SIM card, to implement a call function, a data communication function, and the like. In some embodiments, the device 100 uses an eSIM, namely, an embedded SIM card. The eSIM card may be embedded into the device 100, and cannot be separated from the device 100.

[00114] A software system of the device 100 may use a layered architecture, an event-driven architecture, a microkernel architecture, a micro service architecture, or a cloud architecture. In this embodiment of the present invention, an Android system of a layered architecture is used as an example to illustrate the software structure of the device 100.

[00115] FIG. 2 is a block diagram of a software structure of the device 100 according to an embodiment of the present invention.

[00116] In a layered architecture, software is divided into several layers, and each layer has a clear role and task. The layers communicate with each other through a software interface. In some embodiments, the Android system is divided into four layers: an application layer, an application framework layer, an Android runtime (Android runtime) and system library, and a kernel layer from top to bottom.

[00117] The application layer may include a series of application packages.

[00118] As shown in FIG. 2, an application package may include applications such as Camera, Gallery, Calendar, Phone, Maps, Navigation, WLAN, Bluetooth, Music, Videos, and Messages.

[00119] The application framework layer provides an application programming interface (application programming interface, API) and a programming framework for an application at the application layer. The application framework layer includes some predefined functions.

[00120] As shown in FIG. 2, the application framework layer may include a window manager, a content provider, a view system, a phone manager, a resource manager, a notification manager, and the like.

[00121] The window manager is configured to manage a window program. The window manager may obtain a size of the display, determine whether there is a status bar, perform screen locking, take a screenshot, and the like.

[00122] The content provider is configured to: store and obtain data, and enable the data to be accessed by an application program. The data may include a video, an image, an audio, calls that are made and answered, a browsing history and bookmarks, an address book, and the like.

[00123] The view system includes visual controls such as a control for displaying a text and a control for displaying an image. The view system may be configured to construct an application program. A display interface may include one or more views. For example, a display interface including an SMS message notification icon may include a text display view and an image display view.

[00124] The phone manager is configured to provide a communication function for the device 100, for example, management of a call status (including answering, declining, or the like).

[00125] The resource manager provides various resources such as a localized character string, an icon, an image, a layout file, and a video file for an application program.

[00126] The notification manager enables an application program to display notification information in a status bar, and may be configured to convey a notification message. The notification manager may automatically disappear after a short pause without requiring a user interaction. For example, the notification manager is configured to notify download completion, give a message notification, and the like. The notification manager may alternatively be a notification that appears in a top status bar of the system in a form of a graph or a scroll bar text, for example, a notification of an application that is run on a background, or may be a notification that appears on the screen in a form of a dialog window. For example, text information is displayed in the status bar, an announcement is given, the device vibrates, or the indicator light blinks.

[00127] The Android runtime includes a kernel library and a virtual machine. The Android runtime is responsible for scheduling and management of the Android system.

[00128] The kernel library includes two parts: a function that needs to be called in Java language and a kernel library of Android.

[00129] The application layer and the application framework layer run on the virtual machine. The virtual machine executes java files of the application layer and the application framework layer as binary files. The virtual machine is configured to implement functions such as object lifecycle management, stack management, thread management, security and exception

management, and garbage collection.

[00130] The system library may include a plurality of functional modules, for example, a surface manager (surface manager), a media library (Media Library), a three-dimensional graphics processing library (for example, OpenGL ES), and a 2D graphics engine (for example, SGL).

5 **[00131]** The surface manager is configured to manage a display subsystem and provide fusion of 2D and 3D layers for a plurality of applications.

[00132] The media library supports playback and recording in a plurality of commonly used audio and video formats, and static image files. The media library may support a plurality of audio and video encoding formats, for example, MPEG-4, H.264, MP3, AAC, AMR, JPG, and PNG.

10 **[00133]** The three-dimensional graphics processing library is configured to implement three-dimensional graphics drawing, image rendering, composition, layer processing, and the like.

[00134] The 2D graphics engine is a drawing engine for 2D drawing.

[00135] The kernel layer is a layer between hardware and software. The kernel layer includes at least a display driver, a camera driver, an audio driver, and a sensor driver.

15 **[00136]** The following describes an example of a working process of software and hardware of the device 100 with reference to a photographing scenario.

[00137] When the touch sensor 180K receives a touch operation, a corresponding hardware interrupt is sent to the kernel layer. The kernel layer processes the touch operation into an original input event (including information such as touch coordinates and a time stamp of the touch operation). The original input event is stored at the kernel layer. The application framework layer obtains the original input event from the kernel layer, and identifies a control corresponding to the input event. An example in which the touch operation is a touch operation, and a control corresponding to the touch operation is a control of a camera application icon is used. The camera application invokes an interface of the application framework layer to enable the camera application, then enables the camera driver by invoking the kernel layer, and captures a static image or a video through the camera 193.

20 **[00138]** First, a related technology of the request scheduling method in embodiments of the present invention is explained, to facilitate understanding of a person skilled in the art.

[00139] Transformer (Transformer): a special structure in the deep learning field, including two parts, namely, multi-head self-attention and forward feedback. According to this mechanism, different weights may be allocated based on different importance of parts of input data. This model is mainly used in the fields of natural language processing (NLP) and computer vision (CV).

30 **[00140]** Token (Token): A token corresponds to a word in a text in Chinese natural language processing. An input text is mapped through an analyzer and vocabulary to obtain a corresponding token sequence, and then the corresponding token sequence is sent to the model for further

processing.

[00141] Prefill inference: Generally, a length of input information is large, and a latency is a decisive factor. As a result, a computing bottleneck exists.

[00142] Decode inference: a length of input information is fixed to 1, a delay overhead is low, and a memory access bottleneck exists.

[00143] A throughput (Throughput) is a quantity of requests processed by a system in a unit time. A formula for calculating the throughput in the present invention is as follows:

$$\text{TPS} = \frac{\text{quantity of total requests}}{\text{quantity of chips} \times \text{processing duration}}.$$

[00144] A large-scale language model has set off a new wave in the AI industry. An inference process of a GPT model represented by a generative model is characteristic. Given a group of input token sequences with a length of S, the model outputs an output token sequence with a length of N. Actually, there are N inference processes in this process. That is, the GPT model outputs only one token in each inference process. The output token is concatenated with a previous input token to obtain (S+1, S+2, ..., S+N-1), used as an input in a next inference process until a terminator or a maximum output length is encountered.

[00145] Such a processing manner introduces a problem that a computing amount in each inference process increases accordingly with an increase in a quantity of iterations. In addition, because each iteration includes an input part of a previous iteration, there is a large amount of redundant computation. Therefore, a person skilled in the art uses a KV cache technology to maintain a computing amount of each round of inference after the first round of iteration basically constant, so as to reduce computing power and reduce an inference delay.

[00146] A GPT inference process using the KV cache technology is divided into a first phase, namely, a first-round context filling (Prefill) phase, and a second phase, namely, a multi-round decode (Decode) phase.

[00147] In the prefill phase, an input data dimension is [B, S], where B is a dimension of a group batch, that is, a common batch (batch) in machine learning. A data processing and transmission dimension at an intermediate layer of the model is [B, S, H], where H is a hidden layer dimension. An output data dimension is [B, 1], that is, a quantity of tokens in a batch is B. If there is only one batch, there is only one token, and inference is performed only once. During the inference, key-value KV cache information at the intermediate layer, namely, a transformer (Transformer) layer, is cached in a KV cache, for use in the decoding phase.

[00148] In the decoding phase, an input data dimension is [B, 1]. A data processing and transmission dimension at the intermediate layer of the model is [B, 1, H]. An output data dimension is [B, 1]. The decoding phase is executed N times until a stop condition is met. Similarly,

in an inference process, key-value KV cache information stored in a previous round is first obtained from the KV cache, and then newly generated key-value KV cache information is cached in the KV cache for update.

Table 1 Relationship between computing amount and memory access in the GPT model

S=4K/16K	Prefill phase	Decoding phase
Computing amount (TFLOPS)	572/2816	0.1396/0.1719
Memory access (TB)	0.1853/0.3867	0.1343/0.1504
Ratio of computing amount to memory access (FLOPS/Byte)	3087/7281	1.04/1.14

[00149] Through statistical computing, a relationship between a computing amount and memory access (memory access) in an inference process when $B=1$, $L=88$, and $S=4K/16K$ may be obtained. As shown in Table 1, L is a quantity of transformers at a network layer. It is clear that computing is a decisive factor (a high latency and a computing bottleneck) in the prefill phase, but memory access is a decisive factor (a low latency and a memory access bottleneck) in the decoding phase. When there are a large quantity of service requests from users, group batching is performed on different service requests for inference. A significant difference between the two inference phases causes two problems: 1. Because a length supplement manner is usually used, to be specific, a length of a short input statement in a batch is supplemented to a length of a long input statement, and then inference is performed, there is a large amount of invalid computation in the prefill phase for the short input statement. 2. Short output statements are output in a unified manner after inference of the longest output statement in a batch is complete, resulting in a large amount of invalid computation. These two problems result in a very low throughput of the entire system.

[00150] Based on the foregoing technical problem, an embodiment of the present invention provides a request scheduling system. The present invention mainly improves a throughput of a service system in a large-scale language model service deployment. The large-scale language model is for a model with a transformer structure.

[00151] Currently, a main problem is that there are two inference states in a large model inference process: prefill phase and decoding phase. The significant difference in computing power and memory access in the two phases and characteristics of the two phases are not considered during service deployment. Two phases are all deployed on the same compute node, and when a large quantity of services request services at the same time, service deployment costs are high if a single service request exclusively uses the same compute node. A large amount of redundant computing and latency overheads exist in group batch inference (that is, different services are

grouped into batches and placed on the same compute node), resulting in a low overall throughput of the service system.

[00152] FIG. 3 is a diagram of an architecture of a request scheduling system according to an embodiment of the present invention.

5 **[00153]** The request scheduling system includes a first device, at least one second device, and at least one third device. Both the second device and the third device are connected to the first device. The connection includes a wired connection or a wireless connection.

[00154] For a hardware structure and a software structure of the device provided in this embodiment of the present invention, refer to related descriptions of the device 100 in FIG. 1 and
10 FIG. 2.

[00155] For example, the first device includes a server.

[00156] For example, the second device and the third device each include an electronic device having a computing capability. The electronic device includes a computing card. For example, the computing card includes a GPU or an NPU. The second device is configured to perform prefill
15 inference, that is, inference in a prefill phase. The third device is configured to perform decode inference, that is, inference in a decoding phase.

[00157] For example, the request scheduling system includes the first device, the second device, and the third device. As shown in FIG. 3, the request scheduling system includes a first device 210, a second device 220, and a third device 230.

20 **[00158]** The first device 210 includes a large-scale language model (LLM) service module 211 and a service scheduling module 212.

[00159] The LLM service module 211 is configured to interconnect with an external user service entry.

[00160] The service scheduling module 212 is configured to schedule a new service to enter the
25 second device 220, and schedule an old service that does not meet a stop condition to enter the third device 230.

[00161] The second device 220 includes a first inference component package 221, a first AI engine 222, and a first computing resource module 223.

[00162] The first inference component package 221 and the first AI engine 222 are configured
30 to deploy a large-scale language model and invoke a resource.

[00163] The first computing resource module 223 includes a first batch scheduling module 2231 and a first execution engine 2232.

[00164] The first batch scheduling module 2231 is configured to dynamically group batches or dynamically insert new services. The new service is a service that does not pass through the prefill
35 phase.

[00165] The first execution engine 2232 is configured to perform inference computing on the service requests output by the first batch scheduling module 2231.

[00166] The third device 230 includes a second inference component package 231, a second AI engine 232, and a second computing resource module 233.

5 **[00167]** The second inference component package 231 and the second AI engine 232 are configured to deploy a large-scale language model and invoke a resource.

[00168] The second computing resource module 233 includes a second batch scheduling module 2331 and a second execution engine 2332.

10 **[00169]** The second batch scheduling module 2331 is configured to perform dynamic group batching, dynamic insertion, or dynamic early stopping on an old service that does not meet a stop condition. The old service is a service that passes through the prefill phase.

[00170] The second execution engine 2332 is configured to perform inference computing on the service requests output by the second batch scheduling module 2331.

15 **[00171]** As shown in FIG. 3, the second device 220 further includes a first storage module 241, and the third device 230 further includes a second storage module 242. The second device 220 is connected to the third device 230 in a wired or wireless manner.

[00172] In some possible embodiments, the first storage module 241 is configured to: cache first cache information generated by the second device 220 in an inference process, and transmit the cached first cache information to the second storage module 242. The second execution engine 20 2232 is configured to obtain the first cache information from the second storage module 242 in the inference process.

[00173] In some possible embodiments, the first execution engine 2232 does not send, to the first storage module 241, the first cache information generated by the second device 220 in the inference process, but directly sends the first cache information to the second execution engine 25 2232, so that the second execution engine 2232 uses the first cache information in the inference process.

[00174] The second storage module 242 is further configured to cache cache information generated by the third device 230 in the inference process. The second execution engine 2232 is configured to obtain, from the second storage module 242 in the inference process, cache 30 information generated in a previous inference process.

[00175] For example, the cache information is key-value KV cache information.

[00176] Requests for a plurality of services are scheduled by the service scheduling module 212 to the second device 220 or the third device 230. The second device 220 or the third device 230 performs computing to obtain output information of a current iteration and corresponding cache 35 information, and the output information is returned to the service scheduling module 212. When

the stop condition is not met, the service scheduling module 212 schedules the service to the idle third device 230 to perform a next round of inference. After inference iteration, the service continues to be returned to the service scheduling module 212. When the stop condition is met, the service scheduling module 212 ends scheduling of the service, and provides an external service response.

[00177] In addition, in this embodiment of the present invention, in the request scheduling system, the second device 220, that is, a prefill inference execution container, and the third device 230, that is, a decode inference execution container, are deployed in an independent separate deployment manner. The first device schedules a request, so that the second device and the third device respectively perform inference processes in respective phases. In addition, communication transfer may be performed between the two devices, so that the inference process can be normally performed, and a throughput of the service system can be improved.

[00178] Each of the two execution containers has a batch scheduling module. The batch scheduling module implements two functions: automatic batching and automatic early stopping. Automatic batching is used to dynamically group input information of a plurality of service requests into batches, and includes two manners: group batching and dynamic insertion. Automatic early stopping is used to stop a completed service in advance, release a corresponding resource, and notify the system that the current resource can be inserted and occupied by a next service.

[00179] In the separate deployment manner, a significant difference between a computing power bottleneck in the prefill phase and a memory access bottleneck in the decoding phase is comprehensively considered, so that a throughput of the service system can be improved.

[00180] When there are a large quantity of second devices and third devices in the request scheduling system, the second devices and the third devices are deployed in an independent separate deployment manner. A small batch or a single batch is configured for the second devices, and a large batch is configured for the third devices. A resource proportion of the second device resource to the third device resource may be dynamically adjusted based on system load, to obtain a maximum throughput. The present invention can greatly improve the throughput of the service system.

[00181] Based on the system architecture shown in FIG. 3, an embodiment of the present invention provides a request scheduling method. FIG. 4A and FIG. 4B are a signaling interaction diagram of a request scheduling method according to an embodiment of the present invention. As shown in FIG. 4A and FIG. 4B, the method includes the following steps.

[00182] Step 302: An LLM service module receives a first request from a first user. Before step 302, that is, before a request scheduling system responds to a service, an LLM has been deployed to computing resource modules in the second device 220 and the third device 230 by using an

inference component package and an AI engine, and both the computing resource module and a storage module are ready.

[00183] For example, the first request is a request for a first service.

[00184] For example, the first request includes an identifier of the first service.

5 **[00185]** For example, the identifier includes a service ID.

[00186] For example, the first request does not include requests returned by the second device and the third device. In other words, the first request is a request for a new service.

[00187] In this step, as shown in FIG. 3, the LLM service module receives the first request from the first user.

10 **[00188]** Step 304: The LLM service module sends the first request to a service scheduling module.

[00189] In this step, as shown in FIG. 3, the LLM service module sends the first request to the service scheduling module.

15 **[00190]** Step 306: The service scheduling module sends the first request to a first inference component package.

[00191] In this step, as shown in FIG. 3, the service scheduling module schedules a newly arrived service request to a prefill inference execution container, namely, the second device, and establishes a service pool to store a request for a service that waits due to a resource limitation.

20 **[00192]** Step 308: The first inference component package sends the first request to a first AI engine.

[00193] In this step, as shown in FIG. 3, the first inference component package sends the first request to the first AI engine.

[00194] Step 310: The first AI engine sends the first request to a first computing resource module.

25 **[00195]** In this step, as shown in FIG. 3, the first AI engine sends the first request to the first computing resource module.

[00196] Step 312: The first computing resource module performs group batching on the first request to obtain a first batch, and performs first inference computing on the first batch to obtain a second request and first cache information.

30 **[00197]** In this step, as shown in FIG. 3, a first batch scheduling module of the first computing resource module performs, based on a first computing resource status and a first condition, group batching on the first request and at least one seventh request that arrives at the second device at the same time as the first request, to obtain the first batch. A first execution engine of the first computing resource module performs first inference computing on the first batch, to obtain the
35 second request and the first cache information. The seventh request is a request from the first user,

a second user, or another user.

[00198] For example, the first request further includes input information.

[00199] For example, the input information includes an input token sequence.

5 **[00200]** For example, the first computing resource status includes a quantity of idle resources in the second device.

[00201] For example, the first condition includes a maximum quantity of requests grouped by the second device in a batch .

10 **[00202]** For example, the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request.

[00203] For example, the first output information includes an output token sequence. The first output information is a first output token sequence.

[00204] For example, cache information includes key-value KV cache information. The first cache information is first key-value KV cache information.

15 **[00205]** In some possible embodiments, step 312 specifically includes: The first batch scheduling module in the first computing resource module obtains, based on the first request, the input information corresponding to the first request and a first length of the input information, and performs, based on the first computing resource status, the first condition, and the first length, group batching on the first request and the at least one seventh request that arrives at the second
20 device at the same time as the first request, to obtain a first batch; and the first execution engine of the first computing resource module performs first inference computing on the first batch in a single-batch processing manner, to obtain the second request and the first cache information of the first service. A length of input information corresponding to the seventh request is similar to the first length. This helps reduce a waste of idle resources in a first inference computing process.

25 **[00206]** For example, the first batch includes a plurality of requests.

[00207] For example, the first batch scheduling module groups a plurality of requests that arrive at a same moment into one batch.

[00208] For example, the first inference computing is inference computing in a prefill phase, namely, prefill computing.

30 **[00209]** For example, in a first processing manner, after inference computing of all requests in a current batch ends, computing service resources occupied by the current batch are released, and a next batch is inserted into the idle computing service resources to perform first inference computing. In this embodiment of the present invention, for inference computing in the prefill phase, inference computing is performed based on a single batch. To be specific, after inference
35 computing of all requests in a current batch ends, computing service resources occupied by the

current batch are released, and a next batch is inserted into an idle computing service resource for starting to perform the first inference computing on the next batch.

[00210] For example, when releasing the computing service resource occupied by the current batch, the second device further releases storage resources occupied by the current batch.

5 **[00211]** For example, when the second device has an idle computing service resource, the second device notifies the first device that the second device has the idle computing service resource, and the first device schedules a new service to the second device based on the notification.

[00212] For example, when the second device has no idle computing service resource, the second device notifies the first device that the second device has no idle computing service
10 resource, and the first device suspends, based on the notification, scheduling a new service to the second device.

[00213] Step 314: The first computing resource module sends the first cache information to a second computing resource module.

[00214] In this step, as shown in FIG. 3, the first execution engine of the first computing
15 resource module sends the first cache information to the second computing resource module.

[00215] Optionally, as shown in FIG. 3, the first computing resource module may send the first cache information to a first storage module, the first storage module sends the first cache information to a second storage module, and then the second storage module sends the first cache information to a second execution engine module. The second storage module may automatically
20 send the first cache information to the second execution engine module, or may send the first cache information to the second execution engine module based on a message sent by the second execution engine module.

[00216] Optionally, as shown in FIG. 3, the first computing resource module may directly send the first cache information to the second computing resource module.

25 **[00217]** Step 316: The first computing resource module sends the second request to the service scheduling module. 318 or 320 continues to be performed.

[00218] In this step, as shown in FIG. 3 and FIG. 4A and FIG. 4B, the first computing resource module sends the second request to the service scheduling module.

[00219] Step 318: When the first output information meets a specified condition, the service
30 scheduling module sends the first output information to the LLM service module. The LLM service module sends the first output information to the first user, and a process ends.

[00220] In this embodiment of the present invention, the first device determines whether the output information of the service meets the specified condition. If the specified condition is not met, the first device continues to perform scheduling and inference computing on the service. If
35 the specified condition is met, the first device performs early stopping on the service, and sends

second output information to the first user.

[00221] Step 320: When the first output information does not meet a specified condition, the service scheduling module schedules the second request to a second inference component package.

[00222] In this step, as shown in FIG. 3, the service scheduling module sends the second request
5 to the second inference component package.

[00223] For example, the second request is a request for the first service returned by the second device.

[00224] For example, the specified condition includes at least one of the following:
a length of output information is greater than or equal to a first threshold; or
10 output information includes a terminator.

[00225] Step 322: The second inference component package sends the second request to a second AI engine.

[00226] In this step, as shown in FIG. 3, the second inference component package sends the second request to the second AI engine.

[00227] Step 324: The second AI engine sends the second request to the second computing
15 resource module.

[00228] In this step, as shown in FIG. 3, the second AI engine sends the second request to the second computing resource module.

[00229] Step 326: The second computing resource module performs group batching on the
20 second request to obtain a second batch.

[00230] In this step, as shown in FIG. 3, a second batch scheduling module of the second computing resource module performs group batching on the second request based on a second computing resource status and a second condition to obtain the second batch.

[00231] In some possible embodiments, step 326 specifically includes: The second batch
25 scheduling module performs, based on the second computing resource status and the second condition, group batching on the second request and at least one eighth request that arrives at the third device at the same time as the second request, to obtain the second batch, where the eighth request is from the second device or the third device.

[00232] For example, the second computing resource status includes a quantity of idle resources
30 in the third device.

[00233] For example, the second condition includes a maximum quantity of batches grouped by the third device.

[00234] For example, the second batch includes a plurality of requests.

[00235] For example, the second batch scheduling module groups at least one request that
35 arrives at a same moment into one batch. In the decoding phase, an input length of each request is

1. Provided that there is an idle position in the current batch, a token whose length is 1 can be placed in the batch for inference.

[00236] Step 328: The second computing resource module performs second inference computing on the second batch based on the first cache information, to obtain a third request and second cache information.

[00237] In this step, as shown in FIG. 3, the second execution engine of the second computing resource module performs second inference computing on the second batch based on the first cache information, to obtain the third request and the second cache information.

[00238] In some possible embodiments, step 328 specifically includes: The second execution engine of the second computing resource module performs second inference computing on the second batch in a second processing manner based on the first cache information, to obtain the third request and the second cache information of the first service.

[00239] For example, the third request includes the identifier of the first service and the second output information corresponding to the first service, and the third request is associated with the second request.

[00240] For example, the second cache information is cache information required when the third device performs second inference computing on the third request.

[00241] For example, the second inference computing is inference computing in the decoding phase, namely, decode computing.

[00242] For example, in a second processing manner, if there is a target request whose output information meets the specified condition in a current batch inference process, a computing service resource of the target request is released, and a request in a next batch is inserted into the idle computing service resource to perform second inference computing. In this embodiment of the present invention, for inference computing in the decoding phase, inference computing is performed based on a plurality of batches. To be specific, in the current batch inference process, if a service ends inference in advance, that is, there is a target request whose output information meets the specified condition, a computing service resource of the target request may be released in advance, and before the current batch ends, a request in a next batch is inserted into the idle computing service resource to perform the second inference computing. Therefore, compared with the first processing manner in which the resources of the current batch are released only after inference of all the requests in the current batch ends, the second processing manner in which the resources of the current batch are released without waiting for inference of all the requests in the current batch to be ended, so that no idle resource is wasted, and a system throughput is further improved.

[00243] For example, when releasing the computing service resource occupied by the target

request, the third device further releases a storage resource occupied by the target request.

[00244] For example, when the third device has an idle computing service resource, the third device notifies the first device that the third device has the idle computing service resource, and the first device schedules an old service to the second device based on the notification.

5 **[00245]** For example, when the third device has no idle computing service resource, the third device notifies the first device that the third device has no idle computing service resource, and the first device suspends, based on the notification, scheduling the old service to the third device.

[00246] Step 330: The second computing resource module sends the second cache information to the second storage module.

10 **[00247]** In this step, as shown in FIG. 3, the second execution engine of the second computing resource module sends the second cache information to the second storage module.

[00248] Step 332: The second computing resource module sends the third request to the service scheduling module. Step 334 or step 336 continues to be performed.

[00249] In this step, as shown in FIG. 3, the second execution engine of the second computing
15 resource module sends the third request to the service scheduling module.

[00250] For example, the third request is a request for the first service returned by the third device.

[00251] Step 334: When the second output information meets the specified condition, the service scheduling module sends the second output information to the LLM service module. The
20 LLM service module sends the second output information to the first user, and a process ends.

[00252] In this embodiment of the present invention, the first device determines whether the output information of the service meets the specified condition. If the specified condition is not met, the first device continues to perform scheduling and inference computing on the service. If the specified condition is met, the first device performs early stopping on the service, and sends
25 the second output information to the first user. After early stopping the service, the first device may indicate the third device to release a resource corresponding to the service. The third device reports a current resource status of the third device to the first device. If there is an idle computing service resource in the current resource status, the first device continues to schedule a service to the third device to insert the idle computing service resource, to implement dynamic insertion.

30 **[00253]** In this step, as shown in FIG. 3, when the second output information meets the specified condition, the service scheduling module sends the second output information to the LLM service module. The LLM service module sends the second output information to the first user, and a process ends.

[00254] Step 336: When the second output information does not meet the specified condition,
35 the service scheduling module schedules the third request to the second inference component

package.

[00255] In this step, as shown in FIG. 3, when the second output information does not meet the specified condition, the service scheduling module sends the third request to the second inference component package, so that the third device continues to perform a next round of decode computing on the third request, and uses, in the next round of decode computing, the second cache information generated in a previous round of decode computing. The second execution engine sends cache information generated in the next round of decode computing to the second storage module for storage, for use of the second execution engine when performing a still next round of decode computing.

[00256] For example, the request scheduling system includes four second devices and one third device. FIG. 5 is a diagram of prefill inference and decode inference according to an embodiment of the present invention. As shown in FIG. 5, the four second devices are a device 0, a device 1, a device 2, and a device 3, and one third device is a device x. FIG. 5 shows an inference process in which a first processing manner is used for prefill inference and a second processing manner is used for decode inference. To be specific, a row in prefill inference on the left side indicates that a group of computing resources are used to process a request for one service, and there are four groups of prefill computing resources and requests for four services in total; and four rows in the decode inference on the right side indicate that a group of computing resources are used to process requests of services whose quantity is 4 in a batch, and there are one group of computing resources and requests for four services in total. In FIG. 5, each grid represents a computing latency. Prefill computing on the left side and decode computing on the right side are performed in parallel. There is no time sequence relationship.

[00257] A process of the prefill inference and the decode inference shown in FIG. 5 is described below.

[00258] (1) Lengths of input token sequences of the four services req1, req2, req3, and req4 are respectively 3, 2, 4, and 3. Therefore, when the four services arrive at the device 0, the device 1, the device 2, and the device 3 at the same time, computing of req2 is complete in advance, followed by req1, req4, and req3.

[00259] (2) In the prefill computing, the req2 that completes computing in advance enters the decode computing in advance. A diagram of a system startup process is shown on the right side of FIG. 5. The device x needs to form one batch, and the batch includes four requests. In this case, the four services req1, req2, req3, and req4 all need to enter the decode computing. As shown in FIG. 5, in the decode inference on the right side, req2 arrives first, then req1 and req4 arrive, and finally req3 arrives. Therefore, req1 and req4 generate one idle bubble (bubble), and req3 generates two idle bubbles. The bubble does not exist again after the system becomes stable.

[00260] (3) In the decode computing, a quantity of output tokens of req1 to req4 is 5, 7, 6, and 4, respectively. Therefore, although req4 is the second to enter the decode computing in the prefill computing, req4 is the first to complete the decode computing, and output information of req4 is the first one that meets the specified condition in the batch. For req4 that completes the decode computing, the computing is stopped in advance (early stopping mechanism) and corresponding occupied resources are released. Then, the computing of req1, req2, and req3 is complete.

[00261] (4) In the decode computing process of req1 to req4, req5 to req8 have reached a prefill inference execution container for computing, and the computing has been completed before the computing of req1 to req4 is complete. Therefore, in a process in which req4, req1, req2, and req3 successively stop the computing, the service scheduling module successively and automatically schedules req6, req5, req7, and req8 to the device x, and the device x successively and respectively inserts req6, req5, req7, and req8 into resource positions occupied by req4, req1, req2, and req3. Subsequent computing is performed after dynamic insertion.

[00262] (5) After the system is stable, steps (3) and (4) are repeated for dynamic insertion and group batching, dynamic stop, and providing a service response.

[00263] It should be noted that, when scheduling the third request for the first service to the third device, the service scheduling module schedules only the third request to the third device that stores cache information generated through the first service in a previous round of decode inference. Therefore, in the decoding phase, a decode part is a service iterated in the current third device, and a request for the service is not forwarded to another third device.

[00264] In conclusion, in the present invention, the execution engine sends, to the service scheduling module, the request that is output through the inference computing, and the service scheduling module determines that the request meets the specified condition, so as to implement automatic early stopping of the service; and the batch scheduling module performs group batching on the request to obtain a batch, so as to implement group batching. When the execution engine has the idle computing service resource, the batch scheduling module automatically sends the request in the next batch to the execution engine for dynamic insertion.

[00265] In this embodiment of the present invention, a request for a new service from a user is scheduled to the second device, that is, the prefill inference execution container performs inference computing in the prefill phase; and a request for an old service that passes through the prefill phase is scheduled to the third device, that is, a decode inference execution container performs inference computing in the decoding phase. In the present invention, in a separate deployment manner, a significant difference between a computing power bottleneck in the prefill phase and a memory access bottleneck in the decoding phase is comprehensively considered, so that the throughput of the service system can be improved.

[00266] Optionally, as shown in FIG. 6A and FIG. 6B, the method further includes step 338 to step 372.

[00267] Step 338: The LLM service module receives a fourth request from the second user. Before step 338, that is, before the request scheduling system responds to the service, the LLM has been deployed to the computing resource modules in the second device and the third device by using the inference component package and the AI engine, and both the computing resource module and the storage module are ready.

[00268] For example, the second user is different from the first user. The first device can schedule requests from different users.

[00269] For example, the fourth request is a request for the second service.

[00270] For example, the fourth request includes an identifier of the second service.

[00271] For example, the identifier includes a service ID.

[00272] For example, the fourth request does not include requests returned by the second device and the third device. In other words, the fourth request is a request for a new service.

[00273] In this step, as shown in FIG. 3, the LLM service module receives the fourth request from the second user.

[00274] Step 340: The LLM service module sends the fourth request to the service scheduling module.

[00275] In this step, as shown in FIG. 3, the LLM service module sends the fourth request to the service scheduling module.

[00276] Step 342: The service scheduling module sends the fourth request to the first inference component package.

[00277] In this step, as shown in FIG. 3, the service scheduling module schedules a newly arrived service request to the prefill inference execution container, namely, the second device, and establishes a service pool to store a request for a service that waits due to a resource limitation.

[00278] Step 344: The first inference component package sends the fourth request to the first AI engine.

[00279] In this step, as shown in FIG. 3, the first inference component package sends the fourth request to the first AI engine.

[00280] Step 346: The first AI engine sends the fourth request to the first computing resource module.

[00281] In this step, as shown in FIG. 3, the first AI engine sends the fourth request to the first computing resource module.

[00282] Step 348: The first computing resource module performs group batching on the fourth request to obtain a third batch, and performs first inference computing on the third batch to obtain

a fifth request and third cache information.

[00283] In this step, as shown in FIG. 3, the first batch scheduling module of the first computing resource module performs group batching on the fourth request based on the first computing resource status and the first condition to obtain the third batch. The first execution engine of the first computing resource module performs first inference computing on the third batch to obtain the fifth request and the third cache information.

[00284] For example, the fourth request further includes input information.

[00285] For example, the input information includes an input token sequence.

[00286] For example, the first computing resource status includes a quantity of idle resources in the second device.

[00287] For example, the first condition includes a maximum quantity of requests grouped by the second device in a batch.

[00288] For example, the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request.

[00289] For example, the third output information includes an output token sequence. The third output information is a third output token sequence.

[00290] For example, cache information includes key-value KV cache information. The third cache information is third key-value KV cache information.

[00291] In some possible embodiments, step 348 specifically includes: The first batch scheduling module in the first computing resource module obtains, based on the fourth request, input information corresponding to the fourth request and a second length of the input information, and performs group batching on the fourth request based on the first computing resource status, the first condition, and the second length, to obtain a third batch; and the first execution engine of the first computing resource module performs first inference computing on the third batch in the first processing manner, to obtain the fifth request and the third cache information of the second service.

[00292] For example, the third batch includes a plurality of requests.

[00293] For example, the first batch scheduling module groups a plurality of requests that arrive at a same moment into one batch.

[00294] For example, the first inference computing is inference computing in the prefill phase, namely, prefill computing.

[00295] For example, in a first processing manner, after inference computing of all requests in a current batch ends, computing service resources occupied by the current batch are released, and a next batch is inserted into the idle computing service resources to perform first inference

computing.

[00296] Step 350: The first computing resource module sends the third cache information to the second computing resource module.

[00297] In this step, as shown in FIG. 3, the first execution engine of the first computing resource module sends the third cache information to the second computing resource module.

[00298] Optionally, as shown in FIG. 3, the first computing resource module may send the third cache information to the first storage module, the first storage module sends the third cache information to the second storage module, and then the second storage module sends the third cache information to the second execution engine module. The second storage module may automatically send the third cache information to the second execution engine module, or may send the third cache information to the second execution engine module based on a message sent by the second execution engine module.

[00299] Optionally, as shown in FIG. 3, the first computing resource module may directly send the third cache information to the second computing resource module.

[00300] Step 352: The first computing resource module sends the fifth request to the service scheduling module. Step 354 or step 356 continues to be performed.

[00301] In this step, as shown in FIG. 3 and FIG. 4A and FIG. 4B, the first computing resource module sends the fifth request to the service scheduling module.

[00302] Step 354: When the third output information meets the specified condition, the service scheduling module sends the third output information to the LLM service module. The LLM service module sends the third output information to the second user, and a process ends.

[00303] Step 356: When the third output information does not meet the specified condition, the service scheduling module schedules the fifth request to the second inference component package.

[00304] In this step, as shown in FIG. 3, the service scheduling module sends the fifth request to the second inference component package.

[00305] For example, the fifth request is a request for the second service returned by the second device.

[00306] For example, the specified condition includes at least one of the following:
a length of output information is greater than or equal to a first threshold; or
output information includes a terminator.

[00307] Step 358: The second inference component package sends the fifth request to the second AI engine.

[00308] In this step, as shown in FIG. 3, the second inference component package sends the fifth request to the second AI engine.

[00309] Step 360: The second AI engine sends the fifth request to the second computing

resource module.

[00310] In this step, as shown in FIG. 3, the second AI engine sends the fifth request to the second computing resource module.

5 **[00311]** Step 362: The second computing resource module performs group batching on the fifth request to obtain a fourth batch.

[00312] In this step, as shown in FIG. 3, the second batch scheduling module of the second computing resource module performs group batching on the fifth request based on the second computing resource status and the second condition to obtain the fourth batch.

10 **[00313]** In some possible embodiments, step 362 specifically includes: The second batch scheduling module performs group batching on the fifth request based on the second computing resource status and the second condition, to obtain the fourth batch.

[00314] For example, the second computing resource status includes a quantity of idle resources in the third device.

15 **[00315]** For example, the second condition includes a maximum quantity of batches grouped by the third device.

[00316] For example, the fourth batch includes a plurality of requests.

20 **[00317]** For example, the second batch scheduling module groups at least one request that arrives at a same moment into one batch. In the decoding phase, an input length of each request is 1. Provided that there is an idle position in the current batch, a token whose length is 1 can be placed in the batch for inference.

[00318] Step 364: The second computing resource module performs second inference computing on the fourth batch based on the third cache information, to obtain a sixth request and fourth cache information.

25 **[00319]** In this step, as shown in FIG. 3, the second execution engine of the second computing resource module performs second inference computing on the fourth batch based on the third cache information, to obtain the sixth request and the fourth cache information.

30 **[00320]** In some possible embodiments, step 364 specifically includes: The second execution engine of the second computing resource module performs second inference computing on the fourth batch in the second processing manner based on the third cache information, to obtain the sixth request and the fourth cache information of the second service.

[00321] For example, the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request.

35 **[00322]** For example, the fourth cache information is cache information required when the third device performs inference computing on the sixth request.

[00323] For example, the second inference computing is inference computing in the decoding phase, namely, decode computing.

[00324] For example, in the second processing manner, if there is a target request whose output information meets the specified condition in a current batch inference process, a computing service resource of the target request is released, and a request in a next batch is inserted into the idle computing service resource to perform second inference computing.

[00325] Step 366: The second computing resource module sends the fourth cache information to the second storage module.

[00326] In this step, as shown in FIG. 3, the second execution engine of the second computing resource module sends the fourth cache information to the second storage module.

[00327] Step 368: The second computing resource module sends the sixth request to the service scheduling module. Step 370 or step 372 continues to be performed.

[00328] In this step, as shown in FIG. 3, the second execution engine of the second computing resource module sends the sixth request to the service scheduling module.

[00329] For example, the sixth request is a request for the second service returned by the third device.

[00330] Step 370: When the fourth output information meets the specified condition, the service scheduling module sends the fourth output information to the LLM service module. The LLM service module sends the fourth output information to the second user, and a process ends.

[00331] In this step, as shown in FIG. 3, when the fourth output information meets the specified condition, the service scheduling module sends the fourth output information to the LLM service module. The LLM service module sends the fourth output information to the second user, and a process ends.

[00332] Step 372: When the fourth output information does not meet the specified condition, the service scheduling module schedules the sixth request to the second inference component package.

[00333] Based on the system architecture shown in FIG. 3, an embodiment of the present invention provides a request scheduling method. FIG. 7A and FIG. 7B are a flowchart of a request scheduling method according to an embodiment of the present invention. As shown in FIG. 7A and FIG. 7B, the method includes the following steps.

[00334] Step 402: A first device receives a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service.

[00335] For example, the identifier includes a service ID.

[00336] For example, the first request does not include requests returned by the second device and the third device. In other words, the first request is a request for a new service.

[00337] Step 404: The first device schedules the first request to the second device.

[00338] The first device schedules a newly arrived service request to a prefill inference execution container, namely, the second device, and establishes a service pool to store a request for a service that waits due to a resource limitation.

5 **[00339]** Step 406: The second device sends a second request to the first device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request.

[00340] For example, the first output information includes an output token sequence. The first output information is a first output token sequence.

10 **[00341]** Step 408: The first device schedules the second request to the third device when the first output information does not meet a specified condition, where the third device is different from the second device.

[00342] For example, the second request is a request for the first service returned by the second device.

15 **[00343]** For example, the specified condition includes at least one of the following:
a length of output information is greater than or equal to a first threshold; or
output information includes a terminator.

[00344] Step 410: The third device sends a third request to the first device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request.

20 **[00345]** Step 412: The first device schedules the third request to the third device when the second output information does not meet the specified condition.

[00346] For example, the third request is a request for the first service returned by the third device.

25 **[00347]** The first device sends the third request to the third device, so that the third device continues to perform a next round of decode computing on the third request, and uses, in a next round of decode computing, second cache information generated in a previous round of decode computing. The third device stores cache information generated in the next round of decode computing, for use in a still next round of decode computing.

30 **[00348]** Optionally, as shown in FIG. 7A and FIG. 7B, between step 404 and step 406, step 4052 to step 4054 are further included.

[00349] Step 4052: The second device performs group batching on the first request based on a first computing resource status and a first condition to obtain a first batch.

[00350] For example, the first request further includes input information.

35 **[00351]** For example, the input information includes an input token sequence.

[00352] For example, the first computing resource status includes a quantity of idle resources in the second device.

[00353] For example, the first condition includes a maximum quantity of requests grouped by the second device in a batch.

5 **[00354]** In some possible embodiments, step 4052 specifically includes: The second device obtains, based on the first request, the input information corresponding to the first request and a first length of the input information; and performs, based on the first computing resource status, the first condition, and the first length, group batching on the first request and the at least one
10 seventh request that arrives at the second device at the same time as the first request, to obtain the first batch.

[00355] The seventh request is a request from the first user, a second user, or another user.

[00356] A length of input information corresponding to the seventh request is similar to the first length. This helps reduce a waste of idle resources in a first inference computing process.

[00357] Step 4054: The second device performs first inference computing on the first batch to
15 obtain the second request and first cache information.

[00358] For example, cache information includes key-value KV cache information. The first cache information is first key-value KV cache information.

[00359] In some possible embodiments, step 4052 specifically includes: The second device performs first inference computing on the first batch in the first processing manner.

20 **[00360]** For example, the first inference computing is inference computing in a prefill phase, namely, prefill computing.

[00361] For example, in the first processing manner, after inference computing of all requests in a current batch ends, computing service resources occupied by the current batch are released, and a next batch is inserted into the idle computing service resources to perform first inference
25 computing.

[00362] In this embodiment of the present invention, after step 4052, the method further includes: The second device sends the first cache information to the third device.

[00363] Optionally, after step 406, the method further includes: The first device sends the first output information to the first user when the first output information meets the specified condition.

30 **[00364]** Optionally, as shown in FIG. 7A and FIG. 7B, between step 408 and step 410, step 4092 to step 4094 are further included.

[00365] Step 4092: The third device performs group batching on the second request based on a second computing resource status and a second condition to obtain a second batch.

[00366] For example, the second computing resource status includes a quantity of idle resources
35 in the third device.

[00367] For example, the second condition includes a maximum quantity of batches grouped by the third device.

[00368] In some possible embodiments, step 4092 specifically includes: performing, based on the second computing resource status and the second condition, group batching on the second request and at least one eighth request that arrives at the third device at the same time as the second request, to obtain the second batch, where the eighth request is from the second device or the third device.

[00369] Step 4094: The third device performs second inference computing on the second batch based on the first cache information, to obtain the third request and the second cache information.

[00370] For example, the second inference computing is inference computing in the decoding phase, namely, decode computing. In the decoding phase, the input length of each request is 1. Provided that there is an idle position in the current batch, a token whose length is 1 can be placed in the batch for inference.

[00371] In some possible embodiments, step 4094 specifically includes: The third device performs second inference computing on the second batch in the second processing manner based on the first cache information, to obtain the third request and the second cache information.

[00372] For example, in the second processing manner, if there is a target request whose output information meets the specified condition in the current batch inference process, a computing service resource of the target request is released, and a request in a next batch is inserted into the idle computing service resource to perform second inference computing.

[00373] In this embodiment of the present invention, after step 4094, the method further includes: The third device stores the second cache information.

[00374] For example, the second cache information is cache information required when the third device performs second inference computing on the third request.

[00375] Optionally, as shown in FIG. 7A and FIG. 7B, after step 412, the method further includes step 414 to step 424.

[00376] Step 414: The first device receives a fourth request from a second user, where the fourth request is a request for a second service, and the fourth request includes an identifier of the second service.

[00377] Step 416: The first device schedules the fourth request to the second device.

[00378] Step 418: The second device sends a fifth request to the first device, where the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request.

[00379] Step 420: The first device schedules the fifth request to the third device when the third output information does not meet the specified condition.

- [00380]** Step 422: The third device sends a sixth request to the first device, where the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request.
- [00381]** Step 424: The first device schedules the sixth request to the third device when the fourth
5 output information does not meet the specified condition.
- [00382]** Optionally, between step 416 and step 418, step 4172 to step 4174 are further included.
- [00383]** Step 4172: The second device performs group batching on the fourth requests based on the first computing resource status and the first condition to obtain a third batch.
- [00384]** For example, the fourth request further includes input information.
- 10 **[00385]** In some possible embodiments, step 4172 specifically includes: The second device obtains, based on the fourth request, the input information corresponding to the fourth request and a second length of the input information; and performs group batching on the fourth request based on the first computing resource status, the first condition, and the second length, to obtain the third batch.
- 15 **[00386]** Step 4174: The second device performs first inference computing on the third batch to obtain a fifth request and third cache information.
- [00387]** In some possible embodiments, step 4172 specifically includes: The second device performs first inference computing on the third batch in the first processing manner.
- [00388]** For example, in the first processing manner, after inference computing of all requests
20 in a current batch ends, computing service resources occupied by the current batch are released, and a next batch is inserted into the idle computing service resources to perform first inference computing.
- [00389]** In this embodiment of the present invention, after step 4172, the method further includes: The second device sends the third cache information to the third device.
- 25 **[00390]** Optionally, after step 418, the method further includes: The first device sends the third output information to the first user when the third output information meets the specified condition.
- [00391]** Optionally, between step 420 and step 422, step 4212 to step 4214 are further included.
- [00392]** Step 4212: The third device performs group batching on the fifth request based on a second computing resource status and a second condition to obtain a fourth batch.
- 30 **[00393]** Step 4214: The third device performs second inference computing on the fourth batch based on third cache information, to obtain a sixth request and fourth cache information.
- [00394]** In some possible embodiments, step 4214 specifically includes: The third device performs second inference computing on the fourth batch in the second processing manner based on the third cache information, to obtain the sixth request and the fourth cache information.
- 35 **[00395]** For example, in the second processing manner, if there is a target request whose output

information meets the specified condition in the current batch inference process, a computing service resource of the target request is released, and a request in a next batch is inserted into the idle computing service resource to perform second inference computing.

[00396] In this embodiment of the present invention, after step 4094, the method further includes: The third device stores the fourth cache information.

[00397] The present invention discloses a request scheduling method and a device. The method includes: A first device receives a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service; the first device schedules the first request to a second device; the second device sends a second request to the first device, where the second request includes the identifier of the first service and first output information, and the second request is associated with the first request; when the first output information does not meet a specified condition, the first device schedules the second request to a third device, where the third device is different from the second device; the third device sends a third request to the first device, where the third request includes the identifier of the first service and second output information, and the third request is associated with the second request; and when the second output information does not meet the specified condition, the first device schedules the third request to the third device, so that a throughput of a service system in a large-scale language model service deployment can be improved.

[00398] Based on the system architecture shown in FIG. 3, an embodiment of the present invention provides a request scheduling method, used in a first device. FIG. 8 is a flowchart of another request scheduling method according to an embodiment of the present invention. As shown in FIG. 8, the method includes the following steps.

[00399] Step 502: A first device receives a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service.

[00400] Step 504: Schedule the first request to a second device.

[00401] Step 506: Receive a second request sent by the second device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request.

[00402] Step 508: Schedule the second request to a third device when the first output information does not meet a specified condition, where the third device is different from the second device.

[00403] Step 510: Receive a third request sent by the third device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request.

[00404] Step 512: Schedule the third request to the third device when the second output

information does not meet the specified condition.

[00405] In the technical solution of the request scheduling method provided in this embodiment of the present invention, the method includes: The first device receives at least one first request, where the first request is a request for a service; schedules the at least one first request to the second device, where the first request does not include requests returned by the second device and the third device; receives at least one second request returned by the second device, and schedules the at least one second request to the third device; receives at least one third request returned by the third device, determines, based on the third request, whether the third request meets a third condition, and if determining that the third request does not meet the third condition, schedules the third request to the third device, so that a throughput of a service system in a large-scale language model service deployment can be improved.

[00406] FIG. 9 is a diagram of a structure of a first device according to an embodiment of the present invention. It should be understood that the first device 600 can perform the steps of the first device in the foregoing request scheduling method. To avoid repetition, details are not described herein again. The first device 600 includes a first transceiver unit 601.

[00407] The first transceiver unit 601 is configured to: receive a first request from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service; schedule the first request to a second device; receive a second request sent by the second device, where the second request includes the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request; schedule the second request to a third device when the first output information does not meet a specified condition, where the third device is different from the second device; receive a third request sent by the third device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request; and when the second output information does not meet the specified condition, schedule the third request to the third device.

[00408] Optionally, the first transceiver unit 601 is further configured to: receive a fourth request from a second user, where the fourth request is a request for a second service, and the fourth request includes an identifier of the second service; schedule the fourth request to the second device; receive a fifth request sent by the second device, where the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request; schedule the fifth request to the third device when the third output information does not meet the specified condition; receive a sixth request sent by the third device, where the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is

associated with the fifth request; and schedule the sixth request to the third device when the fourth output information does not meet the specified condition.

[00409] Optionally, the first transceiver unit 601 is further configured to: when the second output information meets the specified condition, send the second output information to the first user.

[00410] Optionally, the identifier includes a service ID.

[00411] Optionally, the specified condition includes at least one of the following:
a length of output information is greater than or equal to a first threshold; or
the output information includes a terminator.

[00412] FIG. 10 is a diagram of a structure of a second device according to an embodiment of the present invention. It should be understood that the second device 700 can perform steps of the second device in the foregoing request scheduling method. To avoid repetition, details are not described herein again. The second device 700 includes a second transceiver unit 701 and a second processing unit 702.

[00413] The second transceiver unit 701 is configured to receive a first request sent by a first device from a first user, where the first request is a request for a first service, and the first request includes an identifier of the first service; and send a second request to the first device, where the second request includes an identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request.

[00414] Optionally, the second transceiver unit 701 is further configured to receive a fourth request sent by the first device from a second user, where the fourth request is a request for a second service, and the fourth request includes an identifier of the second service; and send a fifth request to the first device, where the fifth request includes the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request.

[00415] Optionally, before the second transceiver unit 701 sends the second request to the first device, the second processing unit 702 is configured to perform, based on a first computing resource status and a first condition, group batching on the first request and at least one seventh request that arrives at the second device at the same time as the first request, to obtain a first batch, where the seventh request is a request from the first user, the second user, or another user; and perform first inference computing on the first batch to obtain the second request and first cache information.

[00416] Optionally, the second transceiver unit 701 is further configured to send the first cache information to a third device.

[00417] Optionally, the cache information includes key-value KV cache information.

[00418] Optionally, the second processing unit 702 is specifically configured to obtain, based on the first request, input information corresponding to the first request and a first length of the input information; and perform, based on the first computing resource status, the first condition, and the first length, group batching on the first request and the at least one seventh request that arrives at the second device at the same time as the first request, to obtain the first batch.

[00419] FIG. 11 is a diagram of a structure of a third device according to an embodiment of the present invention. It should be understood that the third device 800 can perform steps of the third device in the foregoing request scheduling method. To avoid repetition, details are not described herein again. The third device 800 includes a third transceiver unit 801 and a third processing unit 802.

[00420] The third transceiver unit 801 is configured to receive a second request sent by a first device from a second device, where the second request includes an identifier of a first service and first output information corresponding to the first service, the second request is associated with the first request, and the first request is a request for the first service from a first user; and send a third request to the first device, where the third request includes the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request.

[00421] Optionally, the third transceiver unit 801 is further configured to receive a fifth request sent by the first device from the second device, where the fifth request includes an identifier of a second service and third output information corresponding to the second service, the fifth request is associated with a fourth request, and the fourth request is a request for the second service from a second user; and send a sixth request to the first device, where the sixth request includes the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request.

[00422] Optionally, before the third transceiver unit 801 sends the third request to the first device, the third processing unit 802 is configured to perform, based on a second computing resource status and a second condition, group batching on the second request and at least one eighth request that arrives at the third device at the same time as the second request, to obtain a second batch, where the eighth request is from the second device or the third device.

[00423] The third transceiver unit 801 is further configured to receive first cache information sent by the second device.

[00424] The third processing unit 802 is further configured to perform second inference computing on the second batch based on the first cache information, to obtain the third request and second cache information.

[00425] It should be understood that the first device 600, the second device 700, and the third

device 800 herein are embodied in a form of functional units. The term "unit" herein may be implemented in a form of software and/or hardware. This is not specifically limited. For example, the "unit" may be a software program, a hardware circuit, or a combination thereof that implements the foregoing function. The hardware circuit may include an application-specific integrated circuit (application-specific integrated circuit, ASIC), an electronic circuit, a processor (for example, a shared processor, a dedicated processor, or a group processor) configured to execute one or more software or firmware programs, a memory, a merged logic circuit, and/or another appropriate component that supports the described function.

[00426] Therefore, the units in the examples described in embodiments of the present invention can be implemented by using electronic hardware, or a combination of computer software and electronic hardware. Whether the functions are performed by hardware or software depends on particular applications and design constraint conditions of the technical solutions. A person skilled in the art may use different methods to implement the described functions for each particular application, but it should not be considered that the implementation goes beyond the scope of the present invention.

[00427] An embodiment of this application provides a device. The device may be a terminal device or a circuit device built into the terminal device. The device may be configured to perform functions/steps in the foregoing method embodiments.

[00428] An embodiment of this application provides a readable storage medium. The readable storage medium stores instructions. When the instructions are run on a device or a processor, the device is enabled to perform the functions/steps in the foregoing method embodiments.

[00429] An embodiment of this application further provides a program product including instructions. When the program product runs on a device or at least one processor, the device is enabled to perform the functions/steps in the foregoing method embodiments.

[00430] In embodiments of this application, "at least one" refers to one or more, and "a plurality of" refers to two or more. For example, A and/or B may represent the following three cases: Only A exists, both A and B exist, and only B exists. A and B may be singular or plural. The character "/" generally indicates an "or" relationship between the associated objects. "At least one of the following" and similar expressions refer to any combination of these terms, including any combination of single or plural terms. For example, at least one of a, b, and c may represent: a, b, c, a-b, a-c, b-c, or a-b-c, where a, b, and c may be one or more.

[00431] A person of ordinary skill in the art may be aware that the units and algorithm steps described in the embodiments disclosed in this specification can be implemented by a combination of electronic hardware, computer software, and electronic hardware. Whether the functions are performed by hardware or software depends on particular applications and design constraint

conditions of the technical solutions. A person skilled in the art may use different methods to implement the described functions for each particular application, but it should not be considered that the implementation goes beyond the scope of this application.

[00432] It may be clearly understood by a person skilled in the art that, for the purpose of convenient and brief description, for a detailed working process of the foregoing system, apparatus, and unit, refer to a corresponding process in the foregoing method embodiments. Details are not described herein again.

[00433] In several embodiments provided in this application, when any function is implemented in a form of a software functional unit and sold or used as an independent product, the function may be stored on a computer-readable storage medium. Based on such an understanding, the technical solutions of this application essentially, or the part contributing to the current technology, or some of the technical solutions may be implemented in a form of a software product. The computer software product is stored in a storage medium, and includes several instructions for indicating a device (which may be a personal computer, a server, or a network device) to perform all or a part of the steps of the methods described in embodiments of this application. The foregoing storage medium includes any medium that can store program code, such as a USB flash drive, a removable hard disk, a read-only memory (read-only memory, ROM), a random access memory (random access memory, RAM), a magnetic disk, or an optical disc.

[00434] The foregoing descriptions are merely specific implementations of embodiments of this application. Any variation or replacement readily figured out by a person skilled in the art within the technical scope disclosed in this application shall fall within the protection scope in embodiments of this application. Therefore, the protection scope of this application shall be subject to the protection scope of the claims.

CLAIMS

1. A request scheduling method, applied to a request scheduling system, wherein the method comprises:

receiving, by a first device, a first request from a first user, wherein the first request is a request for a first service, and the first request comprises an identifier of the first service;

scheduling, by the first device, the first request to a second device;

sending, by the second device, a second request to the first device, wherein the second request comprises the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request;

scheduling, by the first device, the second request to a third device when the first output information does not meet a specified condition, wherein the third device is different from the second device;

sending, by the third device, a third request to the first device, wherein the third request comprises the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request; and

scheduling, by the first device, the third request to the third device when the second output information does not meet the specified condition.

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

receiving, by the first device, a fourth request from a second user, wherein the fourth request is a request for a second service, and the fourth request comprises an identifier of the second service;

scheduling, by the first device, the fourth request to the second device;

sending, by the second device, a fifth request to the first device, wherein the fifth request comprises the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request;

scheduling, by the first device, the fifth request to the third device when the third output information does not meet the specified condition;

sending, by the third device, a sixth request to the first device, wherein the sixth request comprises the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request; and

scheduling, by the first device, the sixth request to the third device when the fourth output information does not meet the specified condition.

3. The method according to claim 1 or 2, wherein the method further comprises:

sending, by the first device, the second output information to the first user when the second

output information meets the specified condition.

4. The method according to any one of claims 1 to 3, wherein the request scheduling system comprises the first device, at least one second device, and at least one third device.

5. A request scheduling method, used in a first device, wherein the method comprises:

5 receiving a first request from a first user, wherein the first request is a request for a first service, and the first request comprises an identifier of the first service;

scheduling the first request to a second device;

receiving a second request sent by the second device, wherein the second request comprises the identifier of the first service and first output information corresponding to the first service, and
10 the second request is associated with the first request;

scheduling the second request to a third device when the first output information does not meet a specified condition, wherein the third device is different from the second device;

receiving a third request sent by the third device, wherein the third request comprises the identifier of the first service and second output information corresponding to the first service, and
15 the third request is associated with the second request; and

scheduling the third request to the third device when the second output information does not meet the specified condition.

6. The method according to claim 5, wherein the method further comprises:

receiving a fourth request from a second user, wherein the fourth request is a request for a
20 second service, and the fourth request comprises an identifier of the second service;

scheduling the fourth request to the second device;

receiving a fifth request sent by the second device, wherein the fifth request comprises the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request;

25 scheduling the fifth request to the third device when the third output information does not meet the specified condition;

receiving a sixth request sent by the third device, wherein the sixth request comprises the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request; and

30 scheduling the sixth request to the third device when the fourth output information does not meet the specified condition.

7. The method according to claim 5, wherein the method further comprises:

sending the second output information to the first user when the second output information meets the specified condition.

35 8. The method according to any one of claims 5 to 7, wherein the identifier comprises a

service ID.

9. The method according to any one of claims 5 to 8, wherein the specified condition comprises at least one of the following:

a length of output information is greater than or equal to a first threshold; or

5 output information comprises a terminator.

10. A request scheduling method, used in a second device, wherein the method comprises:

receiving a first request sent by a first device from a first user, wherein the first request is a request for a first service, and the first request comprises an identifier of the first service; and

10 sending a second request to the first device, wherein the second request comprises the identifier of the first service and first output information corresponding to the first service, and the second request is associated with the first request.

11. The method according to claim 10, wherein the method further comprises:

15 receiving a fourth request sent by the first device from a second user, wherein the fourth request is a request for a second service, and the fourth request comprises an identifier of the second service; and

sending a fifth request to the first device, wherein the fifth request comprises the identifier of the second service and third output information corresponding to the second service, and the fifth request is associated with the fourth request.

20 12. The method according to claim 10, wherein before the sending a second request to the first device, the method further comprises:

performing, based on a first computing resource status and a first condition, group batching on the first request and at least one seventh request that arrives at the second device at the same time as the first request, to obtain a first batch, wherein the seventh request is a request from the first user, a second user, or another user; and

25 performing first inference computing on the first batch to obtain the second request and first cache information.

13. The method according to claim 12, wherein the method further comprises:

sending the first cache information to the third device.

30 14. The method according to claim 12 or 13, wherein the cache information comprises key-value KV cache information.

15. The method according to any one of claims 12 to 14, wherein the performing, based on a first computing resource status and a first condition, group batching on the first request and at least one seventh request that arrives at the second device at the same time as the first request, to obtain a first batch comprises:

35 obtaining, based on the first request, input information corresponding to the first request and

a first length of the input information; and

performing group batching on the first request and the at least one seventh request based on the first computing resource status, the first condition, and the first length, to obtain the first batch.

16. A request scheduling method, used in a third device, wherein the method comprises:

5 receiving a second request sent by a first device from a second device, wherein the second request comprises an identifier of a first service and first output information corresponding to the first service, the second request is associated with a first request, and the first request is a request for the first service from a first user; and

10 sending a third request to the first device, wherein the third request comprises the identifier of the first service and second output information corresponding to the first service, and the third request is associated with the second request.

17. The method according to claim 16, wherein the method further comprises:

15 receiving a fifth request sent by the first device from a second device, wherein the fifth request comprises an identifier of a second service and third output information corresponding to the second service, the fifth request is associated with a fourth request, and the fourth request is a request for the second service from a second user; and

20 sending a sixth request to the first device, wherein the sixth request comprises the identifier of the second service and fourth output information corresponding to the second service, and the sixth request is associated with the fifth request.

18. The method according to claim 16 or 17, wherein before the sending a third request to the first device, the method further comprises:

25 performing, based on a second computing resource status and a second condition, group batching on the second request and at least one eighth request that arrives at the third device at the same time as the second request, to obtain a second batch, wherein the eighth request is from the second device or the third device;

receiving first cache information sent by the second device; and

performing second inference computing on the second batch based on the first cache information, to obtain the third request and second cache information.

30 19. A device, comprising a processor and a memory, wherein the memory is configured to store a computer program, the computer program comprises program instructions, and when the processor runs the program instructions, the device is enabled to perform the steps of the method according to any one of claims 5 to 9, the steps of the method according to any one of claims 10 to 15, or the steps of the method according to any one of claims 16 to 18.

35 20. A readable storage medium, wherein the readable storage medium stores a program, the program comprises program instructions, and when the program request is run by a device, the

device is enabled to perform the method according to any one of claims 5 to 9, or the method according to any one of claims 10 to 15, or the method according to any one of claims 16 to 18.