SOFTWARE DEVELOPMENT METHOD AND SYSTEM

TECHNICAL FIELD

[0001] This application relates to the field of computer technologies, and more specifically, to a software development method and system.

BACKGROUND

[0002] With the development of information technologies, software development scenarios become increasingly diversified. For example, software development may involve a plurality of programming languages, operating systems, or hardware architectures. Differences in development scenarios imply variations in the entire development compilation chain (or development environment). As a result, software development has placed increasingly high requirements on development environments and development tools.

[0003] An integrated development environment (integrated development environment, IDE) is an application used to provide program development environments. Different IDE versions vary in their support for development scenarios. For example, software development may require the use of a plurality of programming languages, which may need to be supported by different IDE versions. In this case, a user needs to select a plurality of IDE versions to achieve support for the corresponding programming languages. For another example, software development may involve different operating systems, and an IDE version is typically bound to an operating system and a chip architecture. In this case, the user needs to select a plurality of IDE versions to support software development in the corresponding operating systems. In the foregoing process, when developing software, the user needs to configure various hardware and/or software according to the development scenarios, while also managing software development processes and the like in different IDEs. This makes it difficult for the user to focus on the implementation of business logic for software development, severely affecting the development efficiency of the user and thus degrading the development experience of the user.

SUMMARY

[0004] This application provides a software development method and system, to help improve development experience of a user.

[0005] According to a first aspect, a software development method is provided, applied in an

integrated development environment IDE. The IDE includes an IDE frontend and a first IDE backend, the first IDE backend supports running in a plurality of types of environments, the plurality of types of environments include at least one of the following: a plurality of types of operating systems or a plurality of types of hardware architectures, and the method includes: The IDE frontend obtains user configuration information, where the configuration information includes at least one of the following: a hardware architecture type or an operating system type. The IDE frontend deploys the first IDE backend in a first environment based on the configuration information, so that software development on the first IDE backend in the first environment is implemented by using the IDE frontend, where the first environment is an environment determined based on the configuration information in the plurality of types of environments supported by the first IDE backend.

[0006] According to the solution in this embodiment of this application, the IDE is divided into the IDE frontend and an IDE backend (for example, the first IDE backend). A specific operation of software development may be completed by the first IDE backend. A same IDE backend supports running in a plurality of types of environments, which facilitates support for a plurality of development scenarios and improves flexibility of a development mode. In this way, the user can select a proper development scenario, a proper hardware architecture, a proper operating system, and the like according to a requirement to perform software development, without considering an IDE version problem or manually configuring or installing a development environment. In this case, the user can focus on service logic of software development, thereby helping improve development efficiency and improving development experience of the user.

[0007] For example, the IDE frontend may be deployed in a local system.

[0008] For example, there may be one or more first IDE backends. In other words, the IDE frontend may interact with one or more IDE backends.

[0009] The first environment may include any one or more of the following: a local container, a local virtual machine, a remote container, or a remote virtual machine.

[0010] With reference to the first aspect, in some implementations of the first aspect, the IDE frontend runs in a client program of the IDE or a client program of a browser.

[0011] According to the solution in this embodiment of this application, the IDE frontend may be deployed in the client program of the IDE, or may be deployed in the client program of the browser, which provides the user with a plurality of selection solutions, thereby further improving flexibility of the development mode, helping further improve development efficiency, and further improving development experience of the user.

[0012] With reference to the first aspect, in some implementations of the first aspect, the IDE frontend includes a user interface UI component and an adaptation layer, the UI component is

configured to provide a user interface, and the adaptation layer is configured to provide a unified interface for the UI component, so that the UI component runs in one or more client programs.

[0013] According to the solution in this embodiment of this application, the adaptation layer may provide the unified interface to the UI component, and shield a difference between API interfaces of different client programs, to implement interaction between the UI component and a desktop application framework layer. In this way, a frontend UI running in different client programs/frameworks can be implemented based on a same set of UI component code. In other words, the frontend UI may support adaptation of a same set of UI component code by using a cross-platform desktop framework, to run in a plurality of client programs and/or frameworks.

[0014] With reference to the first aspect, in some implementations of the first aspect, the first IDE backend includes an IDE kernel, and the IDE kernel is configured to provide language features of a plurality of programming languages.

[0015] For example, the language feature may include any one or more of the following: code completion, browsing, navigation, inspection, compilation and building, testing, debugging, or refactoring.

[0016] According to the solution in this embodiment of this application, the IDE backend may use a unified IDE kernel, and a same IDE kernel may support a plurality of programming languages. In this way, the user may select a proper programming language to perform software development according to a requirement, without considering an IDE version problem, thereby further improving flexibility of a development mode, helping further improve development efficiency, and further improving development experience of the user.

[0017] With reference to the first aspect, in some implementations of the first aspect, the IDE kernel includes: an indexing system, configured to store index information of source code; a language parsing adaptation layer, configured to invoke a language parser corresponding to a programming language of the source code to parse the source code; and a language feature layer, configured to provide language features of a plurality of programming languages, where the plurality of programming languages include the programming language of the source code.

[0018] According to the solution in this embodiment of this application, the IDE backend may use a unified IDE kernel, and the IDE kernel includes a language-dependent layer and a language-independent layer. The language-dependent layer, namely, the language feature layer, may provide the language features of the plurality of programming languages. The indexing system and the language parsing adaptation layer at the language-independent layer are language-independent, and may be applied to language features of all programming languages. In this way, a same IDE kernel can support a plurality of programming languages.

[0019] With reference to the first aspect, in some implementations of the first aspect, the

plurality of types of environments include a local system in which the IDE frontend is located and/or another environment different from the local system.

[0020] With reference to the first aspect, in some implementations of the first aspect, the first environment includes one or more instances, and that the IDE frontend deploys the first IDE backend in the first environment based on the configuration information includes: The IDE frontend obtains the one or more instances based on the configuration information, where the first IDE backend is deployed in the one or more instances; or the IDE frontend obtains the one or more instances based on the configuration information; and when the first IDE backend is not deployed in a part or all of the one or more instances, the IDE frontend deploys the first IDE backend in the instance in which the first IDE backend is not deployed. With reference to the first aspect, in some implementations of the first aspect, the method further includes: The IDE frontend starts the first IDE backend in a first instance in the one or more instances, so that the first IDE backend in the first instance provides a language feature of a first programming language supported by the first instance.

[0021] With reference to the first aspect, in some implementations of the first aspect, the method further includes: The IDE frontend establishes a persistent connection to the first IDE backend in the first instance.

[0022] In this way, the first IDE backend in the first instance can be in a standby state at any time, thereby improving interaction efficiency.

[0023] With reference to the first aspect, in some implementations of the first aspect, that the IDE frontend starts the first IDE backend in the first instance in the one or more instances includes: The IDE frontend starts the first IDE backend in the first instance based on a first operation. The first operation indicates the first programming language.

[0024] With reference to the first aspect, in some implementations of the first aspect, the first operation includes an open operation of a first code file, and the first programming language is a programming language of the first code file.

[0025] With reference to the first aspect, in some implementations of the first aspect, the configuration information is indicated by a login request carrying verification information.

[0026] With reference to the first aspect, in some implementations of the first aspect, when the first IDE backend runs in another environment different from the local system, the IDE frontend and the first IDE backend communicate with each other through a secure shell SSH tunnel.

[0027] With reference to the first aspect, in some implementations of the first aspect, the method further includes: The IDE frontend sends a first request to the first IDE backend in the first environment; the first IDE backend in the first environment generates a first response to the first request; and the first IDE backend in the first environment returns the first response to the IDE

frontend.

[0028] The IDE frontend cooperates with the first IDE backend in the first environment, to implement software development in a system in which the first IDE backend is located.

[0029] With reference to the first aspect, in some implementations of the first aspect, the IDE further includes a second IDE backend running in a second environment, and the method further includes: The IDE frontend sends a second request to the first IDE backend in the first environment; the first IDE backend in the first environment forwards the second request to the second IDE backend in the second environment; the second IDE backend in the second environment generates a second response to the second request; the second IDE backend in the second environment sends the second response to the first IDE backend in the first environment; and the first IDE backend in the first environment returns the second response to the IDE frontend.

[0030] The first IDE backend and the second IDE backend may run in different environments. In other words, the first environment and the second environment may be of different types.

[0031] The IDE frontend, the first IDE backend, and the second IDE backend cooperate to implement software development in a system in which the second IDE backend is located.

[0032] With reference to the first aspect, in some implementations of the first aspect, the method further includes: The first IDE backend in the first environment projects, via a display server protocol, a graphical user interface program running in a remote environment onto the local system in which the IDE frontend is located.

[0033] In this way, a graphical user interface of a process in the remote environment can be output to an operating system in which the IDE frontend is located.

[0034] According to a second aspect, a software development system is provided, including an IDE frontend and a first IDE backend. The first IDE backend supports running in a plurality of types of environments, and the plurality of types of environments include at least one of the following: a plurality of types of operating systems or a plurality of types of hardware architectures. The IDE frontend is configured to: obtain user configuration information, where the configuration information includes at least one of the following: a hardware architecture type or an operating system type; and deploy the first IDE backend in a first environment based on the configuration information, so that software development on the first IDE backend in the first environment is implemented by using the IDE frontend, where the first environment is an environment determined based on the configuration information in the plurality of types of environments supported by the first IDE backend.

[0035] According to the solution in this embodiment of this application, the IDE is divided into the IDE frontend and an IDE backend (for example, the first IDE backend). A specific operation of software development may be completed by the first IDE backend. A same IDE

backend supports running in a plurality of types of environments, which facilitates support for a plurality of development scenarios and improves flexibility of a development mode. In this way, the user can select a proper development scenario, a proper hardware architecture, a proper operating system, and the like according to a requirement to perform software development, without considering an IDE version problem or manually configuring or installing a development environment. In this case, the user can focus on service logic of software development, thereby helping improve development efficiency and improving development experience of the user.

[0036] With reference to the second aspect, in some implementations of the second aspect, the IDE frontend runs in a client program of an IDE or a client program of a browser.

[0037] With reference to the second aspect, in some implementations of the second aspect, the IDE frontend includes a user interface UI component and an adaptation layer, the UI component is configured to provide a user interface, and the adaptation layer is configured to provide a unified interface for the UI component, so that the UI component runs in one or more client programs.

[0038] With reference to the second aspect, in some implementations of the second aspect, the first IDE backend includes an IDE kernel, and the IDE kernel is configured to provide language features of a plurality of programming languages.

[0039] With reference to the second aspect, in some implementations of the second aspect, the IDE kernel includes: an indexing system, configured to store index information of source code; a language parsing adaptation layer, configured to invoke a language parser corresponding to a programming language of the source code to parse the source code; and a language feature layer, configured to provide language features of a plurality of programming languages, where the plurality of programming languages include the programming language of the source code.

[0040] With reference to the second aspect, in some implementations of the second aspect, the plurality of types of environments include a local system in which the IDE frontend is located and/or another environment different from the local system.

[0041] With reference to the second aspect, in some implementations of the second aspect, the first environment includes one or more instances, and the IDE frontend is specifically configured to: obtain the one or more instances based on the configuration information, where the first IDE backend is deployed in the one or more instances; or obtain the one or more instances based on the configuration information; and when the first IDE backend is not deployed in a part or all of the one or more instances, deploy the first IDE backend in the instance in which the first IDE backend is not deployed.

[0042] With reference to the second aspect, in some implementations of the second aspect, the IDE frontend is further configured to start the first IDE backend in a first instance in the one or more instances, so that the first IDE backend in the first instance provides a language feature of a

first programming language supported by the first instance.

[0043] With reference to the second aspect, in some implementations of the second aspect, the IDE frontend is further configured to establish a persistent connection to the first IDE backend in the first instance.

[0044] With reference to the second aspect, in some implementations of the second aspect, the IDE frontend is specifically configured to start the first IDE backend in the first instance based on a first operation. The first operation indicates the first programming language.

[0045] With reference to the second aspect, in some implementations of the second aspect, the first operation includes an open operation of a first code file, and the first programming language is a programming language of the first code file.

[0046] With reference to the second aspect, in some implementations of the second aspect, the configuration information is indicated by a login request carrying verification information.

[0047] With reference to the second aspect, in some implementations of the second aspect, the IDE frontend is further configured to send a first request to the first IDE backend in the first environment; and the first IDE backend in the first environment is configured to: generate a first response to the first request; and return the first response to the IDE frontend.

[0048] With reference to the second aspect, in some implementations of the second aspect, the system further includes a second IDE backend running in the second environment, and the IDE frontend is further configured to send a second request to the first IDE backend in the first environment; the first IDE backend in the first environment is configured to forward the second request to the second IDE backend in the second environment; the second IDE backend in the second environment is configured to: generate a second response to the second request; and send the second response to the first IDE backend in the first environment; and the first IDE backend in the first environment is further configured to return the second response to the IDE frontend.

[0049] With reference to the second aspect, in some implementations of the second aspect, the first IDE backend in the first environment is further configured to project, via a display server protocol, a graphical user interface program running in a remote environment onto the local system in which the IDE frontend is located.

[0050] It should be understood that extensions to, limitations on, explanations for, and descriptions of related content in the first aspect are also applicable to same content in the second aspect.

[0051] According to a third aspect, a compute device cluster is provided, and includes at least one compute device. Each compute device includes a processor and a memory. The processor of the at least one compute device is configured to execute instructions stored in the memory of the at least one compute device, to enable the compute device cluster to perform the method according

to any one of the first aspect or the implementations of the first aspect.

[0052] According to a fourth aspect, a computer-readable medium is provided, including computer program instructions. When the computer program instructions are executed by a compute device cluster, the compute device cluster performs the method according to any one of the first aspect or the implementations of the first aspect.

[0053] According to a fifth aspect, a computer program product including instructions is provided. When the instructions are run by a compute device cluster, the compute device cluster is enabled to perform the method according to any one of the first aspect or the implementations of the first aspect.

BRIEF DESCRIPTION OF DRAWINGS

[0054] FIG. 1 is a diagram of a structure of an architecture of a heterogeneous operating system;

[0055] FIG. 2 is a diagram of a structure of an IDE architecture according to an embodiment of this application;

[0056] FIG. 3 is a diagram of a structure of a cross-platform desktop framework according to an embodiment of this application;

[0057] FIG. 4A and FIG. 4B are diagrams of structures of another IDE architecture according to an embodiment of this application;

[0058] FIG. 5 is a diagram of a structure of an IDE kernel according to an embodiment of this application;

[0059] FIG. 6 is a diagram of a structure of another IDE architecture according to an embodiment of this application;

[0060] FIG. 7 is a diagram of application projection according to an embodiment of this application;

[0061] FIG. 8A and FIG. 8B are diagrams of application projection in two scenarios according to an embodiment of this application;

[0062] FIG. 9 is a schematic flowchart of a software development method according to an embodiment of this application;

[0063] FIG. 10 is a diagram of applying a solution to a local IDE development scenario according to an embodiment of this application;

[0064] FIG. 11 is a diagram of applying a solution to a web IDE development scenario according to an embodiment of this application;

[0065] FIG. 12 is a diagram of applying a solution to another web IDE development scenario according to an embodiment of this application;

[0066] FIG. 13 is a diagram of applying a solution to a development scenario of remote development of a local IDE according to an embodiment of this application;

[0067] FIG. 14 is a diagram of applying a solution to a development scenario of remote development of a web IDE according to an embodiment of this application;

[0068] FIG. 15 is a diagram of applying a solution to a local IDE development scenario according to an embodiment of this application;

[0069] FIG. 16 is a diagram of applying a solution to a web IDE development scenario according to an embodiment of this application;

[0070] FIG. 17 is a diagram of applying a solution to a development scenario of remote development of a local IDE according to an embodiment of this application;

[0071] FIG. 18 is a flowchart of another software development method according to an embodiment of this application;

[0072] FIG. 19 is a block diagram of a software development system according to an embodiment of this application;

[0073] FIG. 20 is a block diagram of a compute device according to an embodiment of this application;

[0074] FIG. 21 is a block diagram of a compute device cluster according to an embodiment of this application; and

[0075] FIG. 22 is a block diagram of another compute device cluster according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

[0076] The following describes technical solutions of this application with reference to accompanying drawings.

[0077] Terms used in the following embodiments are merely intended to describe specific embodiments, but are not intended to limit this application. The terms "one", "a", and "this" of singular forms used in this specification and the appended claims of this application are intended to include a form like "one or more", unless otherwise specified in the context clearly. It should be further understood that, in the following embodiments of this application, "at least one", "at least one item", and "one or more" mean one, two, or more. "First", "second", and various numbers are merely intended for distinguishing for ease of description, and are not intended to limit the scope of embodiments of this application. "And/or" describes a correspondence between objects that correspond to each other, and indicates that three relationships may exist. For example, "A and/or B" may indicate the following three cases: Only A exists, only B exists, and both A and B exist,

where A and B may be singular or plural. The character "/" generally indicates an "or" relationship between the associated objects. Sequence numbers of the following processes do not mean an execution sequence. The execution sequence of the processes should be determined based on functions and internal logic of the processes, and should not constitute any limitation on an implementation process of embodiments of this application. For example, in embodiments of this application, the terms "301", "401", "501", and the like are merely identifiers for ease of description, and are not intended to limit an execution sequence of steps.

[0078] Reference to "one embodiment", "some embodiments", or the like described in this specification means that one or more embodiments of this application include a specific feature, structure, or characteristic described with reference to embodiments. In this application, the term "example" or "for example" is used to represent giving an example, an illustration, or a description. Any embodiment or design scheme described as an "example" or "for example" in this application should not be explained as being more preferred or having more advantages than another embodiment or design scheme. To be precise, use of the words such as "example" or "for example" is intended to present a relevant concept in a specific manner. The terms "include", "comprise", "have", and their variants all mean "include but are not limited to", unless otherwise specifically emphasized in another manner. In embodiments of this application, descriptions such as "when", "in a case", "assuming that", and "if" all mean that a device performs corresponding processing in a specific objective situation, but are not intended to limit time, do not require that the device has a determining action during implementation, and do not mean any other limitation.

[0079] In this application, "indicating" may include "directly indicating" and "indirectly indicating". When a piece of indication information indicates A, the indication information may directly indicate A or indirectly indicate A, but it does not indicate that the indication information definitely carries A.

[0080] To help a person skilled in the art better understand the technical solutions of this application, the following first describes some terms that may be used in embodiments of this application.

[0081] 1. Integrated development environment (integrated development environment, IDE)

[0082] The IDE is an application that provides a program development environment, usually includes tools such as a code editor, a compiler, a debugger, and a graphical user interface, and is a software development service suite that integrates code writing, analysis, compilation, debugging, and other functions.

[0083] The IDE may include a local IDE and a web (web) IDE.

[0084] The local IDE may be also referred to as a desktop IDE. The local IDE is installed in local devices of users. In other words, developers may install development environments in the

local devices. For example, the local device may be a terminal device, for example, a desktop computer, a laptop computer, or a mobile phone.

[0085] The web IDE is an online IDE, namely, an online integrated development environment for the developers. The web IDE may include an IDE frontend and an IDE backend. The IDE frontend runs in a browser, and the IDE backend may run in a remote environment. For example, the remote environment may be a cloud server. In this way, programmers do not need to install the development environment in the local device; and instead, may perform software development merely by opening the browser. In addition, the web IDE may further support breakpoint debugging, version management, team development, and other functions.

[0086] 2. Heterogeneous system

[0087] The heterogeneous system is a system including different types of hardware or software, and these components may be heterogeneous. The components of the heterogeneous system have respective autonomy, and retain respective application features, complete features, and security features while data sharing is implemented.

[0088] The heterogeneous system may include a plurality of heterogeneous operating systems (operating system, OS) such as HarmonyOS and EulerOS. A part of operating systems may run on different hardware architectures. The hardware architecture may also be referred to as a processor architecture, a chip architecture, a hardware platform, or the like. Different hardware architectures may also be replaced with different processor architectures, different chip architectures, different hardware platforms, different processor types, different hardware types, different chip types, different central processing units (central processing unit, CPU), or the like. HarmonyOS and EulerOS are used as an example. Hardware architectures compatible with these systems include but are not limited to Kunpeng, X86, and ARM. Design and implementation of the heterogeneous operating system feature high performance, scalability, and flexibility.

[0089] FIG. 1 is a diagram of a structure of an architecture of a heterogeneous operating system. As shown in FIG. 1, the architecture of the heterogeneous operating system may include a user interaction layer, an application layer, a system service layer, a heterogeneous operating system kernel subsystem, and a hardware platform.

[0090] The user interaction layer is configured to provide a human-machine interaction interface, for example, a graphical user interface (graphical user interface, GUI) or a command-line interface (command-line interface, CLI).

[0091] The application layer is configured to: provide a multi-language user program framework for an application, and an application programming interface (application programming interface, API) of a multi-language framework for exposing various software and hardware services; and provide the API of the multi-language framework for a device that uses the

system.

[0092] The system service layer includes one or more of the following services: a system basic capability, a basic software service, an enhanced software service, a hardware service, or the like; is configured to provide a basic capability for operations such as running, scheduling, and migration of a distributed application across a plurality of devices of the system; and may include any one or more of the following subsystems: distributed soft bus, distributed data management, distributed task scheduling, multi-language runtime, a common basic library, multi-mode input, graphics, security, AI, or the like.

[0093] The heterogeneous operating system kernel subsystem is configured to provide a basic kernel capability for an upper layer by shielding a multi-kernel difference; may include any one or more of the following: process/thread management, memory management, file system, network management, peripheral management, or the like; and may provide a unified peripheral access capability and driver development and management framework for a hardware ecosystem.

[0094] As shown in FIG. 1, containers based on different operating systems may be created in the systems, for example, a container #1, a container #2, and a container #3 in FIG. 1. The container #1 is a container based on an OS #A, the container #2 is a container based on an OS #B, and the OS #A and the OS #B are different types of operating systems.

[0095] For example, the hardware platform in FIG. 1 may include any one of Kunpeng, X86, or ARM.

[0096] With development of information technologies, software development scenarios become increasingly diversified. This leads to higher requirements of software development for development environments and development tools. Different IDE versions support different development scenarios. A user needs to manually configure and create a proper development environment based on a development scenario before software development.

[0097] The operating system and a hardware architecture are used as an example. The IDE version (version) is usually bound to the operating system and the hardware architecture. In other words, one IDE product corresponds to a plurality of versions, and one version can run only in one operating system and on one hardware architecture. For example, an IDE includes four versions: a version #1, a version #2, a version #3, and a version #4. The four versions respectively run in the following four scenarios: OS #1+hardware type #1, OS #1+hardware type #2, OS #2+hardware type #3, and OS #2+hardware type #4. To perform software development on a hardware architecture, the user needs to install a corresponding IDE version, and create a corresponding development environment. For example, to perform software development based on the hardware type #1, the user needs to install an IDE version that can support the hardware type #1, namely, the version #1.

[0098] A remote development scenario is used as an example. The user starts remote development work in a desktop application, and configures a remote development environment based on the development scenario. Specifically, the following steps may be included:

[0099] (1) An IDE frontend is connected to a remote server through an SSH tunnel. In addition, the user needs to ensure that a to-be-developed operating system has been installed on the remote server.

[00100] For example, to develop an application in an operating system #1, it needs to be ensured that the operating system #1 has been installed on the remote server.

[00101] (2) Configure the development environment.

[00102] The user needs to select a proper IDE version based on the development scenario, to configure the development environment. An IDE of this version is used to run a development project of the user.

[00103] Specifically, the user needs to determine a proper IDE based on a programming language, a chip architecture, and the like.

[00104] (3) Create the development environment.

[00105] After the user performs selection, the system downloads a corresponding IDE backend on the remote server, to create the remote development environment. In other words, the development environment runs as the IDE backend.

[00106] (4) Connect the development environment.

[00107] To be specific, the IDE frontend and the IDE backend are connected. Then, the user may perform remote development.

[00108] It can be learned from the foregoing solution that, when software development is performed in different development scenarios, the user needs to select corresponding hardware, corresponding software, adapted IDE versions, and the like based on different development scenarios. Correspondingly, the user may need to configure and/or install a plurality of types of hardware and software. In addition, the user further needs to manage software development processes, and the like in different IDEs. In this case, the user cannot focus on software development, which seriously affects development efficiency and affects development experience of the user.

[00109] In view of this, embodiments of this application provide a software development system architecture, which can support a plurality of development scenarios such as a plurality of programming languages, a plurality of operating systems, a plurality of hardware architectures, and a plurality of development modes, to help ensure development efficiency of a user and improve development experience of the user.

[00110] In embodiments of this application, the software development system architecture may

also be referred to as an IDE architecture. The IDE architecture in embodiments of this application may be referred to as a "one-kernel multi-mode" architecture. To be specific, a same IDE kernel may support a plurality of development scenarios.

[00111] For example, the plurality of operating systems may include any one or more of the following: EulerOS, HarmonyOS, or another operating system. For example, another operating system may be a Xinchuang operating system.

[00112] For example, the plurality of hardware architectures may include any one or more of the following: Kunpeng, X86, or ARM.

[00113] For detailed descriptions of the plurality of development modes, refer to the following descriptions. Details are not described herein.

[00114] FIG. 2 is a diagram of a structure of an IDE architecture according to an embodiment of this application. The following describes a system architecture in this embodiment of this application with reference to FIG. 2.

[00115] As shown in FIG. 2, a system 200 may include an IDE frontend 210 and a first IDE backend 220. The IDE frontend 210 and the first IDE backend 220 may communicate with each other.

[00116] The term "first" in "first IDE backend" is merely used to indicate that the IDE backend may communicate with the IDE frontend to implement software development, and is not intended to impose any other limitation.

[00117] The IDE frontend is also referred to as a frontend IDE, and the IDE backend is also referred to as a backend IDE.

[00118] The following describes the IDE frontend 210.

[00119] The IDE frontend 210 may be deployed in a local operating system. In this embodiment of this application, the local operating system may also be referred to as a local system. In a possible understanding manner, an operating system in which the IDE frontend 210 is located may be considered as the local operating system.

[00120] The IDE frontend 210 may be configured to provide a user interface (user interface, UI).

[00121] In other words, the IDE frontend 210 may include a frontend UI. For example, the IDE frontend 210 may be a lightweight frontend UI deployed in the local operating system.

[00122] A user may interact with the IDE frontend 210.

[00123] The IDE frontend 210 supports running in a plurality of client programs/frameworks.

[00124] Optionally, the IDE frontend 210 may run in a client program of an IDE. The client program of the IDE may be deployed in the local operating system.

[00125] The user may download the client program of the IDE, install the client program of the

IDE in the local operating system, and run the client program of the IDE to implement running of the IDE frontend 210.

[00126] The client program of the IDE may also be referred to as a client program of a desktop IDE, or a client program of a local IDE.

[00127] Optionally, the IDE frontend 210 may run in a client program of a browser. The client program of the browser may be deployed in the local operating system.

[00128] The user does not need to download the client program of the IDE, and can implement running of the IDE frontend 210 in the browser.

[00129] FIG. 3 is a diagram of a cross-platform desktop framework according to an embodiment of this application. As shown in FIG. 3, an IDE frontend 210 is deployed in the framework.

[00130] As shown in FIG. 3, the framework further includes a desktop application framework layer 310. The desktop application framework layer 310 may include one or more client programs. The one or more client programs are configured to build a cross-platform application. For example, the client programs of the desktop application framework layer 310 may include any one or more of the following: a browser, Electron, Webview2, QT, or Tauri. The browser may be Chromium.

[00131] The one or more client programs may also be replaced with one or more frameworks.

[00132] Different client programs and/or frameworks may provide different application programming interfaces (application programming interface, API).

[00133] In a possible implementation, the IDE frontend 210 may include a UI component 211 and an adaptation layer 212, as shown in FIG. 3.

[00134] The UI component 211 is configured to provide a user interface.

[00135] For example, the UI component 211 may include any one or more of the following: a button, an input box, a menu, an editor, a window, a dialog box, or a terminal (terminal) command line window.

[00136] The adaptation layer 212 is configured to provide a unified interface for the UI component 211, so that the UI component 211 runs in one or more client programs.

[00137] In other words, the adaptation layer 212 is configured to provide a unified interface for the UI component 211, to implement interaction between the UI component and the desktop application framework layer.

[00138] The adaptation layer 212 may also be referred to as a cross-platform desktop framework adaptation layer 212.

[00139] In this way, a frontend UI running in different client programs/frameworks can be implemented based on a same set of UI component code. In other words, the frontend UI may support adaptation of a same set of UI component code by using a cross-platform desktop framework, to run in a plurality of client programs and/or frameworks.

[00140] The adaptation layer 212 may shield a difference between different APIs for the UI component 211, and provide a unified interface for the UI component, so that a same set of UI component code can be adapted to a plurality of client programs and/or frameworks.

[00141] As shown in FIG. 3, the framework may further include a basic runtime environment 320 and an operating system 330. The basic runtime environment 320 is configured to provide a software environment for running a program of a corresponding programming language.

[00142] For example, the basic runtime environment 320 may include any one or more of the following: Node JS, a Python runtime environment, a Java runtime environment, or a low level virtual machine (low level virtual machine, LLVM).

[00143] For example, the operating system 330 may include any one or more of the following: HarmonyOS, EulerOS, or another XC OS.

[00144] The following describes a first IDE backend 220 in this embodiment of this application.

[00145] The first IDE backend 220 may be deployed in a first environment.

[00146] The term "first" in "first environment" is merely used to indicate that the environment is an environment in which the first IDE backend 220 is located, and is not intended to impose any other limitation.

[00147] The first IDE backend 220 may include an IDE kernel. The IDE kernel may be configured to provide a language feature of one or more programming languages.

[00148] The IDE kernel may include a language-dependent layer and a language-independent layer. When the IDE kernel may be configured to provide language features of a plurality of programming languages, the IDE kernel may also be referred to as a multi-language IDE kernel.

[00149] FIG. 4A and FIG. 4B are diagrams of a structure of another IDE architecture according to an embodiment of this application. The solution shown in FIG. 4A and FIG. 4B may be considered as two specific implementations of the solution shown in FIG. 2.

[00150] As shown in FIG. 4A and FIG. 4B, a language-dependent layer may include a language feature layer. A language-independent layer may include a language parsing adaptation layer and an indexing system.

[00151] The language feature layer may be configured to provide language features of a plurality of programming languages.

[00152] For example, the plurality of programming languages may include any one or more of the following: Python, Java, C++, C#, or JavaScript.

[00153] FIG. 5 is a diagram of an IDE kernel according to an embodiment of this application. For example, as shown in FIG. 5, a language feature layer of the IDE kernel includes four language features: a Python language feature, a Java language feature, a C++ language feature, and a C# language feature. It should be understood that the language features in FIG. 5 are merely an

example. In another implementation, the language feature layer may further provide another quantity of language features and/or a language feature of another programming language. This is not limited in this embodiment of this application.

[00154] For example, the language feature may include any one or more of the following: code completion, browsing, navigation, inspection, compilation and building, testing, debugging, or refactoring.

[00155] A language-independent layer may be shared by all language features.

[00156] A language parsing adaptation layer may be configured to invoke a language parser corresponding to a programming language of source code to parse the source code, to generate index information of the source code. The index information of the source code may be provided to the language feature for use.

[00157] An indexing system may be configured to store the index information of the source code.

[00158] A first IDE backend may be further configured to provide another function, for example, store a code model or store the source code.

[00159] Specifically, in a software development process, first source code is used as an example, and the language parsing adaptation layer may invoke a language parser corresponding to a programming language of the first source code to parse the first source code, to generate index information of the first source code for use by a language feature of the programming language of the first source code. The indexing system may store index information of the first source code. The language feature layer may provide the language feature of the programming language of the first source code. In other words, the language feature of the programming language of the first source code at the language feature layer is activated, so that the language feature layer provides the language feature of the programming language.

[00160] For example, the first source code may be source code that is currently being processed, or source code in a currently opened code file.

[00161] There may be one first IDE backend 220, as shown in FIG. 4A; or there may be a plurality of first IDE backends 220, as shown in FIG. 4B. In other words, an IDE frontend may establish a connection to the one first IDE backend 220, or may establish connections to the plurality of first IDE backends 220. In other words, the IDE frontend may interact with the one first IDE backend 220, or may interact with the plurality of first IDE backends 220.

[00162] The plurality of IDE backends 220 may differ in any one or more of the following: an operating system, a hardware architecture, a programming language, or the like.

[00163] For example, a user may need to develop a multi-language project, that is, a project involving a plurality of programming languages. In this case, a corresponding service may be

provided by one first IDE backend 220 for each programming language.

[00164] Two programming languages (for example, Python and Java) are used as an example. The IDE frontend may separately interact with two first IDE backends 220. For example, IDE kernels of the two first IDE backends 220 may be the IDE kernel shown in FIG. 5. When the user performs an operation on a Python file, a language feature layer of one of the first IDE backends 220 provides a language feature of Python, and when the user performs an operation on a Java file, a language feature layer of the other first IDE backend 220 provides a language feature of Java.

[00165] In the solution in this embodiment of this application, the IDE kernel may support a plurality of programming languages, that is, may provide corresponding language services for the plurality of programming languages. When corresponding language services need to be provided for different programming languages, a same IDE kernel may be used for implementation. In this way, the user does not need to select a corresponding IDE version based on a programming language, which helps simplify complexity of use by the user, thereby improving user experience.

[00166] The first IDE backend 220 supports running in a plurality of types of environments. In other words, the first environment may be of a plurality of types.

[00167] The plurality of types of environments include at least one of the following: a plurality of types of operating systems or a plurality of types of hardware architectures.

[00168] In other words, a same IDE backend may be deployed in different operating systems or hardware architectures. In other words, the first IDE backend 220 may support running in a plurality of types of operating systems. The first IDE backend may support running on a plurality of types of hardware architectures.

[00169] In other words, a deployment environment of the first IDE backend 220 is a configurable item. For example, the user may manually configure one or more hardware architecture types and/or one or more operating system types, and a same first IDE backend 220 may be separately deployed in environments configured by the user.

[00170] For example, the operating system is used as an example, and corresponding runtimes are provided for different operating systems, so that the first IDE backend 220 can run in different operating systems. For example, the programming language is Java, the operating system is EulerOS, an operating system of the first environment may be EulerOS, and the first environment provides a Java runtime. In this way, the first IDE backend 220 may run in the first environment.

[00171] The following provides example descriptions of different division manners.

[00172] The first environment may be a local system or a remote environment. In other words, the first IDE backend 220 may be deployed in the local system or the remote environment.

[00173] That the first environment is the local system means that an IDE frontend 210 and the first IDE backend 220 run in a same local system, in other words, in a same operating system. In

a possible understanding manner, any other environment different from the local system in which the IDE frontend 210 is located may be considered as a remote environment.

[00174] For example, the first environment is a remote environment, and the IDE frontend 210 may be further configured to establish a connection to the first environment. If the first IDE backend 220 has been deployed in the first environment, the IDE frontend 210 may be configured to start the first IDE backend 220. If the first IDE backend 220 is not deployed in the first environment, the IDE frontend 210 may deploy the first IDE backend 220 in the first environment and start the first IDE backend 220. For detailed descriptions, refer to the method 600 in the following descriptions. Details are not described herein.

[00175] A system to which the first environment belongs may be a homogeneous system or a heterogeneous system. In the solution of this embodiment of this application, the homogeneous system may also be understood as a homogeneous operating system. In other words, all operating systems in the homogeneous system are a same type of operating system. The heterogeneous system may also be understood as a heterogeneous operating system. In other words, the heterogeneous system includes a plurality of types of operating systems.

[00176] The first environment may include a local compute resource or a remote compute resource.

[00177] In other words, the first IDE backend 220 may be deployed on the local compute resource or the remote compute resource.

[00178] For example, the compute resource may include a container and/or a virtual machine. For example, the first IDE backend 220 may be deployed in any one of the following: a local container, a local virtual machine, a remote container, or a remote virtual machine.

[00179] The foregoing is merely an example, and constitutes no limitation on the solution in this embodiment of this application. In a specific implementation, the first IDE backend 220 may be deployed according to a requirement. For example, the deployment environment of the first IDE backend 220 may be specified by the user. In other words, the first IDE backend 220 may be deployed in an environment specified by the user.

[00180] As described above, there may be one or more first IDE backends 220. When there are a plurality of first IDE backends 220, deployment environments of the plurality of first IDE backends 220 may be of a same type or different types.

[00181] In the solution in this embodiment of this application, the first IDE backend may support a plurality of types of environments, that is, can run in different types of environments. In this way, the user does not need to select a corresponding IDE version based on a development scenario, which helps simplify complexity of use by the user, thereby improving user experience.

[00182] Further, the first IDE backend 220 may further include a backend service proxy (proxy).

The backend service proxy may be configured to: forward a request from the IDE frontend 210 to the IDE kernel for processing, and then return a response to the request to the IDE frontend 210.

[00183] The following provides example descriptions of a software development process by using two manners (Manner #1 and Manner #2) as an example.

[00184] Manner #1:

[00185] In a possible implementation, an IDE frontend 210 and a first IDE backend 220 cooperate with each other to complete software development.

[00186] In Manner #1, the first IDE backend 220 may complete an operation related to software development. In other words, the IDE frontend 210 and the first IDE backend 220 cooperate to implement software development in a system in which the first IDE backend 220 is located.

[00187] Optionally, the IDE frontend 210 may be configured to send a first request to the first IDE backend 220.

[00188] The first IDE backend 220 may be configured to: generate a first response to the first request, and return the first response to the IDE frontend 210.

[00189] The term "first" in "first request" is merely used to indicate that the request is processed by the first IDE backend 220, and is not intended to impose any other limitation. The "first response" is a response to the "first request", and is not intended to impose any other limitation.

[00190] In this development scenario, the IDE frontend 210 sends the request to the first IDE backend 220 for processing. The first IDE backend 220 may return the response to the request to the IDE frontend.

[00191] For example, the request sent by the IDE frontend 210 may be generated based on an operation of a user. In other words, the IDE frontend 210 sends the operation of the user to the first IDE backend 220 in a form of a request. The first IDE backend 220 may process the request, and return a processing result to the IDE frontend 210 in a form of a response.

[00192] For example, the operation of the user may be a code operation of the user, in other words, a code activity, for example, editing, debugging, testing, compilation, and building.

[00193] For example, the code operation of the user may be an "edit" operation. The IDE frontend 210 sends an "edit" request to the first IDE backend 220. An IDE kernel in the first IDE backend 220 may process the request, provide a corresponding operation, a corresponding instruction, or the like, and return the corresponding operation, the corresponding instruction, or the like to the IDE frontend 210.

[00194] For a specific example of the development scenario in Manner #1, refer to FIG. 10 and FIG. 11 in the following descriptions.

[00195] Manner #2:

[00196] FIG. 6 is a diagram of another IDE architecture according to an embodiment of this

application.

[00197] In another possible implementation, a system 200 may further include a second IDE backend 230, as shown in FIG. 6. An IDE frontend 210, a first IDE backend 220, and the second IDE backend 230 cooperate with each other to complete software development.

[00198] The second IDE backend 230 may be deployed in a second environment. The second environment may be a remote environment.

[00199] A first environment and the second environment may be different types of environments. The term "second" in "second environment" is merely used to distinguish from an environment in which the first IDE backend 220 is located, and is not intended to impose any other limitation.

[00200] A system to which the second environment belongs may be a homogeneous system or a heterogeneous system.

[00201] In Manner #2, the second IDE backend 230 may complete an operation related to software development. In other words, the IDE frontend 210, the first IDE backend 220, and the second IDE backend 230 cooperate to implement software development in a system in which the second IDE backend 230 is located.

[00202] In a possible understanding manner, a difference between the first IDE backend 220 and the second IDE backend 230 may be understood as follows: The first IDE backend 220 can communicate with the IDE frontend without forwarding through another IDE backend, whereas the second IDE backend 230 requires forwarding through another IDE backend to communicate with the IDE frontend. The term "second" in "second IDE backend" is merely used to indicate that the IDE backend communicates with the IDE frontend through forwarding by another IDE backend, and is not intended to impose any other limitation.

[00203] The second IDE backend 230 includes a language-dependent layer and a language-independent layer. The foregoing descriptions of the first IDE backend 220 are also applicable to the second IDE backend 230. Details are not described herein again.

[00204] A connection may be established between the first environment and the second environment, so that the IDE frontend 210 and the second IDE backend 230 can communicate through forwarding by the first IDE backend 220.

[00205] Optionally, the IDE frontend 210 may be configured to send a second request to the second IDE backend 230 through the first IDE backend 220. The second IDE backend 230 may be configured to: generate a second response to the second request, and return the second response to the IDE frontend 210 through the first IDE backend 220.

[00206] The term "second" in "second request" is merely used to indicate that the request is processed by the second IDE backend 230, and is not intended to impose any other limitation. The

"second response" is a response to the "second request", and is not intended to impose any other limitation.

[00207] In this development scenario, a request sent by the IDE frontend 210 is forwarded to the second IDE backend 230 through the first IDE backend 220 for processing, and the second IDE backend 230 returns a response to the request to the IDE frontend 210 through the first IDE backend 220.

[00208] For other descriptions, refer to the foregoing descriptions. Details are not described herein again.

[00209] For a specific example of the development scenario in Manner #2, refer to FIG. 13 and FIG. 14 in the following descriptions.

[00210] In the solution in this embodiment of this application, an IDE is divided into an IDE frontend and an IDE backend, the IDE frontend is configured to provide a user interface, and a specific operation of software development may be completed by the IDE backend. A same IDE backend supports running in a plurality of types of environments, which facilitates support for a plurality of development scenarios and improves flexibility of a development mode. In this way, the user can select a proper development scenario, a proper hardware architecture, a proper operating system, and the like according to a requirement to perform software development, without considering an IDE version problem or manually configuring or installing a development environment. In this case, the user can focus on service logic of software development, thereby helping improve development efficiency and improving development experience of the user.

[00211] In addition, in the solution in this embodiment of this application, the IDE frontend may be deployed in a client program of the IDE, or may be deployed in a client program of a browser, which provides the user with a plurality of selection solutions, thereby further improving flexibility of a development mode, helping further improve development efficiency, and further improving development experience of the user.

[00212] In addition, in the solution in this embodiment of this application, the IDE backend may use a unified IDE kernel, and a same IDE kernel may support a plurality of programming languages. In this way, the user may select a proper programming language to perform software development according to a requirement, without considering an IDE version problem, thereby further improving flexibility of a development mode, helping further improve development efficiency, and further improving development experience of the user.

[00213] The IDE backend may use a unified IDE kernel, and the IDE kernel includes a language-dependent layer and a language-independent layer. The language-dependent layer, namely, a language feature layer, may provide language features of a plurality of programming languages. An indexing system and a language parsing adaptation layer at the language-

independent layer are language-independent, and may be applied to language features of all programming languages. In this way, a same IDE kernel can support a plurality of programming languages.

[00214] In the solution in this embodiment of this application, a deployment environment of the IDE frontend, a deployment environment of the IDE backend, and the programming language may all be configurable items. The IDE architecture may support a plurality of development scenarios and a plurality of programming languages, thereby greatly improving flexibility and scalability of a software development manner. The user may flexibly select a proper development scenario and programming language according to a requirement of the user without considering deployment of the development environment, which improves development experience of the user.

[00215] Further, the first IDE backend 220 may be further configured to perform application projection.

[00216] Optionally, the first IDE backend 220 may be further configured to project, via a display server protocol, a GUI program in a third environment onto an operating system (for example, a local system) in which the IDE frontend 210 is located. The third environment is a remote environment.

[00217] The GUI program is projected onto the operating system in which the IDE frontend 210 is located. In other words, a UI of the GUI program is sent to the operating system.

[00218] Application projection may be initiated by the user. For example, the user may be connected to the remote environment by using the client program of the IDE or the client program of the browser, and start a GUI program in the remote environment by using a command line. The first IDE backend 220 may serve as a proxy to send the UI of the GUI program to the client program of the IDE or the client program of the browser based on the display server protocol.

[00219] There may be one or more third environments. There may be one or more GUI programs in one third environment. For example, the first IDE backend 220 may be configured to project, based on the display server protocol, one or more GUI programs in a remote environment onto the operating system (for example, the local system) in which the IDE frontend 210 is located. For another example, the first IDE backend 220 may be configured to project, based on the display server protocol, a plurality of GUI programs in a plurality of remote environments onto the operating system (for example, the local system) in which the IDE frontend 210 is located.

[00220] The term "third" in "third environment" is merely used to describe an environment in which the GUI program is located, and is not intended to impose any other limitation.

[00221] The third environment may be a homogeneous system, or may be a heterogeneous system.

[00222] When the first IDE backend 220 runs in the remote environment, the GUI program may

be a GUI program running in an environment in which the first IDE backend 220 is located, or may be a GUI program running in another environment. This is not limited in this embodiment of this application. For example, the display server protocol may include any one of the following: X, Wayland, or a remote desktop protocol (remote desktop protocol, RDP). For example, the X protocol may be an X11 protocol, namely, an 11^{th} version of the X protocol.

[00223] Specifically, the GUI program in the remote environment may be projected, through an SSH tunnel based on the display server protocol, onto the operating system in which the IDE frontend 210 is located.

[00224] The X protocol is used as an example. The X protocol may include an X server (server) and an X client (client). The X server is configured to: display a screen image, and notify the X client of an input action. The X client is configured to process program logic. The X client is a GUI program. The X server is deployed at the IDE frontend 210. The X server may run in an independent process, and is independent of the projected GUI program. The GUI program can be projected onto the local system through SSH X11 forwarding (forwarding), as shown in FIG. 8A and FIG. 8B. The GUI program in the local system in FIG. 8A and FIG. 8B is a projected GUI program.

[00225] In this way, a graphical user interface of a process in the remote environment can be output to an operating system in which the IDE frontend is located.

[00226] FIG. 7 is a diagram of application projection according to an embodiment of this application. As shown in FIG. 7, the solution in this embodiment of this application may implement application projection. To be specific, a GUI program in a remote environment is projected onto a local system (namely, a local operating system).

[00227] For example, the GUI program in the remote environment may be projected onto the local system through an SSH tunnel based on a display server protocol.

[00228] The remote environment may be a homogeneous system, or may be a heterogeneous system.

[00229] As described above, in some implementations, a first IDE backend 220 may be deployed in the remote environment. In this case, the remote environment in which the GUI program is located and the remote environment in which the first IDE backend 220 is located may be a same remote environment, or may be different remote environments.

[00230] FIG. 8A and FIG. 8B are diagrams of application projection in two scenarios.

[00231] In FIG. 8A, a first environment and a third environment are a same environment. In other words, a first IDE backend 220 and a GUI program are deployed in a same environment. In FIG. 8B, a first environment and a third environment are different environments. In other words, a first IDE backend 220 and a GUI program are deployed in different environments. The two

environments may communicate with each other, to implement application projection.

[00232] FIG. 8A is used as an example. A backend service proxy may forward a request from an IDE frontend 210 to an IDE kernel in the first IDE backend 220 for processing, and then return a response to the request to the IDE frontend 210. The backend service proxy may further send a UI of the GUI program to a local system via a display server protocol.

[00233] It should be understood that FIG. 8A and FIG. 8B are merely examples, and constitute no limitation on the solution in this embodiment of this application. For example, in FIG. 8A and FIG. 8B, projection of only one GUI program in one remote environment is used as an example. In another implementation, another quantity of GUI programs may alternatively be projected, and there may also be another quantity of remote environments in which the GUI program is located.

[00234] The IDE frontend and the IDE backend may communicate with each other in a plurality of manners. A communication manner between the IDE frontend and the IDE backend is related to a specific development mode. For detailed descriptions, refer to the following descriptions. Details are not described herein.

[00235] FIG. 9 is a schematic flowchart of a software development method according to an embodiment of this application. A method 600 shown in FIG. 9 may be performed by the system 200 shown in FIG. 2. An IDE frontend in the method 600 may be the IDE frontend 210 in FIG. 2. A first IDE backend in the method 600 may be the first IDE backend 220 in FIG. 2. For detailed descriptions, refer to the first IDE backend 220.

[00236] As shown in FIG. 9, the method 600 includes the following steps.

[00237] 610: The IDE frontend obtains user configuration information, where the configuration information includes at least one of the following: a hardware architecture type or an operating system type.

[00238] 620: The IDE frontend deploys the first IDE backend in a first environment based on the configuration information, so that software development on the first IDE backend in the first environment is implemented by using the IDE frontend.

[00239] Step 620 may also be understood as follows: The IDE frontend deploys the first IDE backend in the first environment based on the configuration information, so that the user can implement software development on the first IDE backend in the first environment by interacting with the IDE frontend, in other words, can implement software development on the first IDE backend in the first environment by operating the IDE frontend.

[00240] The first environment is a deployment environment of the first IDE backend. A type of the first environment is determined based on the configuration information. In other words, the type of the first environment is indicated by the configuration information. In other words, the first environment is an environment determined based on the configuration information in a plurality

of types of environments supported by the first IDE backend.

[00241] As described above, the deployment environment of the first IDE backend is a configurable item. In step 610, related configuration information of the deployment environment may be obtained, to determine the deployment environment of the first IDE backend. In this way, the first IDE backend is deployed in an environment (for example, the first environment) that conforms to the configuration information.

[00242] The configuration information may include one or more hardware architecture types and/or one or more operating system types. A quantity of configured environment types is not limited in this embodiment of this application.

[00243] In this embodiment of this application, the first IDE backend supports deployment in the plurality of types of environments. For ease of description, the following mainly provides example descriptions of the method 600 by using an example in which the first IDE backend is deployed in a remote environment. A solution in which the first IDE backend is deployed in another environment may be adaptively modified according to a requirement. The first IDE backend may also be replaced with a first IDE backend service.

[00244] Further, the method 600 may further include step 611 (not shown in the figure).

[00245] 611: The IDE frontend obtains the first environment based on the configuration information.

[00246] In step 611, the deployment environment of the first IDE backend, namely, an environment required by a to-be-developed program, may be obtained.

[00247] The first environment may include one or more instances. The one or more instances may also be understood as one or more first environments. In other words, there may be one or more first environments.

[00248] Step 611 may also be understood as that the IDE frontend obtains the one or more instances based on the configuration information. The one or more instances are used to deploy the first IDE backend.

[00249] For example, that the IDE frontend obtains the one or more instances based on the configuration information may include: The IDE frontend creates the one or more instances based on the configuration information.

[00250] Alternatively, that the IDE frontend obtains the one or more instances based on the configuration information may include: The IDE frontend determines the one or more instances based on the configuration information. For example, a cloud service may provide a plurality of candidate instances and allocate the one or more instances to the IDE frontend based on the configuration information.

[00251] Obtaining of the one or more instances may be triggered in a plurality of manners. The

following provides descriptions by using creation of the one or more instances as an example.

[00252] For example, creation of the one or more instances may be triggered by starting of an IDE. Starting of the IDE may also be replaced with starting of the IDE frontend.

[00253] In other words, when the user starts the IDE, creation of the one or more instances is triggered. That the user starts the IDE may be understood as starting the IDE in response to an IDE starting request of the user.

[00254] The user may start a client program of a local IDE to start the IDE frontend. Alternatively, the user may start the IDE frontend in a client program of a browser.

[00255] Alternatively, creation of the one or more instances may be triggered by a login request of the user, in other words, may be triggered by a login operation of the user.

[00256] The login request carries verification information. The login request may be used to request an account of a login user.

[00257] For example, the one or more instances are deployed on a cloud compute service. The user may log in to an account after starting the IDE. The IDE frontend may be connected to the cloud compute service by using this account, and create the one or more instances on the cloud compute service.

[00258] Obtaining of the one or more instances may be alternatively triggered in another manner. This is not limited in this embodiment of this application.

[00259] Types of instances obtained for different users may be the same or different.

[00260] For example, different users may be distinguished based on account information of the users.

[00261] For example, the one or more instances may include any one or more of the following: a container or a virtual machine.

[00262] In a possible implementation, in step 611, a plurality of instances may be obtained. In this case, programming languages corresponding to the plurality of instances may be the same or may be different.

[00263] A programming language corresponding to an instance may be understood as a programming language supported by the instance. In other words, the instance may provide a corresponding runtime environment for the programming language. The instance may also be referred to as an instance of the programming language.

[00264] For example, a programming language corresponding to an instance is Java. In other words, the instance provides a Java runtime. For another example, a programming language corresponding to an instance is Python. In other words, the instance provides a Python runtime.

[00265] Optionally, each of the one or more instances corresponds to a different programming language. In other words, in step 611, instances of one or more programming language may be

obtained.

[00266] The one or more instances may include an instance of a part or all of programming languages supported by the first IDE backend.

[00267] The programming language supported by the first IDE backend may be understood as that the first IDE backend includes a language feature of the programming language.

[00268] For example, the one or more instances include instances of all programming languages supported by the first IDE backend. For example, the first IDE backend supports five programming languages. In step 620, five instances may be created. The five instances respectively correspond to the five programming languages.

[00269] A programming language corresponding to the one or more instances may be determined in a plurality of manners.

[00270] For example, the programming language corresponding to the one or more instances may be preset.

[00271] In an example, the preset programming language may be a default programming language of a system or a programming language preconfigured by the user. For example, default programming languages of the system include Java and Python, or programming languages preconfigured by the user include Java and Python. In step 611, an instance of Java and an instance of Python may be created.

[00272] Alternatively, the programming language corresponding to the one or more instances may be determined based on related information of the user.

[00273] For example, a programming language previously used by the user may be determined based on a historical record of the account of the user, to create a corresponding instance.

[00274] For another example, a project that has been stored by the user may be determined based on the account of the user, and a corresponding instance is created based on a programming language used by the project.

[00275] Alternatively, the programming language corresponding to the one or more instances may be entered by the user.

[00276] For example, after starting the IDE, the user may configure that programming languages that need to be used for current software development include Java and Python, and the configuration information may be carried in a first creation request. In step 611, the instance of Java and the instance of Python may be created.

[00277] The programming language corresponding to the one or more instances may be indicated by the configuration information in step 610, or may be indicated by other information.

[00278] It should be understood that the foregoing is merely an example, and which one or several programming languages for which an instance is to be created may alternatively be

determined in another manner. For example, the programming language corresponding to the one or more instances may be randomly selected. This is not limited in this embodiment of this application.

[00279] A hardware architecture type of the one or more instances may be determined in a plurality of manners.

[00280] In a possible implementation, in step 611, a plurality of instances may be created. In this case, hardware architecture types of the plurality of instances may be the same or may be different.

[00281] For ease of description, the following first provides descriptions by using one hardware architecture type as an example.

[00282] For example, a hardware architecture type of the one or more instances may be preset.

[00283] In an example, the preset hardware architecture type may be a default hardware architecture type of a system or a hardware architecture type preconfigured by the user. For example, the hardware architecture type preconfigured by the user is Kunpeng, and correspondingly, the configuration information may indicate that the hardware architecture type is Kunpeng. In step 611, the one or more instances may be created on hardware of a Kunpeng architecture.

[00284] Alternatively, the hardware architecture type of the one or more instances may be determined based on related information of the user.

[00285] For example, a hardware architecture type previously used by the user may be determined based on a historical record of the account of the user, to create a corresponding instance.

[00286] For another example, a project that has been stored by the user may be determined based on the account of the user, and a corresponding instance is created based on a hardware architecture type used by the project.

[00287] Alternatively, the hardware architecture type of the one or more instances may be entered by the user.

[00288] For example, after starting the IDE, the user may configure that a hardware architecture type that needs to be used for current software development is Kunpeng. Correspondingly, the configuration information may indicate that the hardware architecture type is Kunpeng. In step 611, the one or more instances may be created on hardware of a Kunpeng architecture.

[00289] As described above, the hardware architecture type of the one or more instances may be indicated by the configuration information. A form of the configuration information may be implemented in a plurality of manners. For example, for the hardware architecture type preconfigured by the user, the configuration information may be an identifier of the hardware

architecture type. Alternatively, the configuration information may be an empty field. If the configuration information of the hardware architecture type is "empty", the system creates an instance based on a previously configured hardware architecture type.

[00290] It should be understood that the foregoing is merely an example, and a specifically used hardware architecture may alternatively be determined in another manner. This is not limited in this embodiment of this application.

[00291] As described above, in step 620, a plurality of instances may be created. If a plurality of hardware architecture types are involved, instances need to be created on the plurality of hardware architecture types. For a manner of creating an instance on each hardware architecture, refer to the foregoing descriptions. The following provides descriptions by using an example.

[00292] For example, an instance created on each type of hardware architecture in the plurality of hardware architecture types may include instances of all programming languages supported by the first IDE backend. For example, the first IDE backend supports five programming languages. In step 611, five instances may be created on each type of hardware architecture. The five instances respectively correspond to the five programming languages.

[00293] For example, a programming language corresponding to the instance created on each type of hardware architecture in the plurality of hardware architecture types may be preset.

[00294] For example, default programming languages of the system include Java and Python, or programming languages preconfigured by the user include Java and Python. In step 611, an instance of Java and an instance of Python may be created on each type of hardware architecture.

[00295] For example, a programming language corresponding to the instance created on each type of hardware architecture in the plurality of hardware architecture types may be configured by the user.

[00296] For example, the user may configure that hardware architecture types that need to be used for current software development are Kunpeng and ARM, a programming language of an instance on Kunpeng is Java, and a programming language of an instance on ARM is Python. In step 611, an instance supporting Java may be created on Kunpeng, and an instance supporting Python may be created on ARM.

[00297] The foregoing is merely an example. For another implementation, refer to the foregoing descriptions. Details are not described herein again.

[00298] In a possible implementation, in step 620, a plurality of instances may be created. In this case, operating systems of the plurality of instances may be the same or may be different.

[00299] For ease of description, the following first provides descriptions by using an operating system as an example.

[00300] For example, an operating system type of the one or more instances may be preset.

[00301] In an example, the preset operating system type may be a default operating system type of a system or an operating system type preconfigured by the user. For example, the user preconfigures that the operating system type is EulerOS, and correspondingly, the configuration information may indicate that the operating system type is EulerOS. In step 611, the one or more instances may be created, and an operating system of the one or more instances may be EulerOS.

[00302] Alternatively, an operating system type of the one or more instances may be entered by the user.

[00303] For example, after starting the IDE, the user may configure that an operating system type that needs to be used for current software development is EulerOS, and correspondingly, the configuration information may indicate that the operating system type is EulerOS. In step 611, the one or more instances may be created, and an operating system of the one or more instances may be EulerOS.

[00304] It should be understood that the foregoing is merely an example, and a specifically used operating system type may also be determined in another manner. This is not limited in this embodiment of this application.

[00305] As described above, in step 620, a plurality of instances may be created. If a plurality of operating system types are involved, for a manner of creating an instance based on each operating system, refer to the foregoing descriptions. Programming languages, operating system types, and hardware architecture types corresponding to the plurality of instances may all be set according to a requirement, for example, configured by the user or configured by a system by default.

[00306] For example, the user may need to develop an application of a plurality of programming languages in a plurality of types of operating systems on a plurality of types of hardware architectures. In this case, a plurality of instances that support the plurality of programming languages based on the plurality of types of operating systems may be created on the plurality of types of hardware architectures for the user. For detailed descriptions, refer to the foregoing descriptions. Details are not described herein again. Programming languages, operating system types, and hardware architecture types corresponding to the plurality of instances may all be indicated by the configuration information. Configuration information of different items may be obtained in a same manner or different manners. For example, the operating system type is indicated by the user. In other words, configuration information of the operating system is determined based on an input of the user. For another example, the hardware architecture type is a default type of a system. In other words, configuration information of the hardware architecture may be obtained from the system.

[00307] Optionally, the configuration information is indicated by a login request carrying

verification information. For example, when logging in, the user may configure the hardware architecture type and/or the operating system type. The corresponding configuration information may be indicated by the login request.

[00308] The foregoing is merely an example, and a manner of obtaining the configuration information is not limited in this embodiment of this application.

[00309] Step 620 may include the following cases.

[00310] As described above, the IDE frontend may obtain the one or more instances based on the configuration information. The first IDE backend may have been deployed in the one or more instances, or the first IDE backend may not be deployed.

[00311] If the first IDE backend has been deployed in the one or more instances, a process of obtaining the one or more instances may also be considered as step 620.

[00312] If the first IDE backend is not deployed in a part or all of the one or more instances, step 620 may further include: deploying the first IDE backend in the instance in which the first IDE backend is not deployed.

[00313] For example, the IDE frontend may send an installation package of the first IDE backend to the instance in which the first IDE backend is not deployed, to implement deployment of the first IDE backend in the corresponding instance.

[00314] Further, the method 600 may further include step 630.

[00315] 630: The IDE frontend starts a first IDE backend in a first instance. The first instance belongs to the one or more instances obtained in step 611.

[00316] After the first IDE backend in the first instance is started, a language feature of a first programming language supported by the first instance may be provided.

[00317] In other words, the IDE frontend starts the first IDE backend in the first instance, so that the first IDE backend provides the language feature of the first programming language supported by the first instance.

[00318] In this embodiment of this application, the first IDE backend may be configured to provide language features of a plurality of programming languages. After the first IDE backend is started, the corresponding language features may be activated. Starting a first IDE backend in a corresponding instance may also be considered as a step in activating a corresponding language service.

[00319] In an implementation, in step 630, first IDE backends in all instances may be started.

[00320] In another implementation, in step 630, a first IDE backend in a particular instance (for example, the first instance) may be started.

[00321] Optionally, step 630 may include: starting the first IDE backend in the first instance based on a first operation.

[00322] The first operation may also be considered as an activation operation.

[00323] In an implementation, the first operation may be an operation performed on a first project.

[00324] The first project is any software development project. The term "first" in "first project" is merely for ease of description, and is not intended to impose any other limitation.

[00325] In a possible implementation, the first project may be a project created after the IDE is started this time.

[00326] In this case, the method 600 may further include step 631.

[00327] 631: Create the first project.

[00328] For example, the first project may be created based on a project template. For example, the first project may be a multi-language project, and the user may create the first project by using a corresponding template, of the multi-language project, provided by the system. That the user creates the first project may be understood as creating the first project in response to a project creation request of the user, where the project creation request indicates to create the first project.

[00329] In another possible implementation, the first project may be a stored project opened after the IDE is started this time.

[00330] In this case, the method 600 may further include step 632.

[00331] 632: Open the first project.

[00332] After starting the IDE, the user may open the previously stored first project. That the user opens the first project may be understood as opening the first project in response to a project opening request of the user, where the project opening request indicates to open the first project.

[00333] The language service is provided by the first IDE backend. In the solution in this embodiment of this application, language services of different programming languages are provided by a unified first IDE backend. In other words, the language services of the different programming languages are provided by IDE kernels of a same architecture. For detailed descriptions of the first IDE backend, refer to FIG. 2. Details are not described herein again.

[00334] The language service may also be referred to as a language server, and is configured to provide programming support, in other words, is configured to support language features such as code completion, code formatting, code diagnosis, navigation, hovering, highlighting, and type inference. That the first IDE backend is configured to provide the language features of the plurality of programming languages may also be replaced with that the first IDE backend is configured to provide language services of the plurality of programming languages.

[00335] For example, the language service may provide programming support, for example, provide a group of operations or instructions, based on a current state of the IDE frontend, for example, a current position of a cursor in the IDE frontend or a current hovering position of a

mouse.

[00336] A language service of a programming language may be configured to provide programming support for the programming language, namely, programming support provided when software development is performed based on the programming language.

[00337] The language service and the IDE frontend may communicate with each other through JavaScript (JS) object notation (JavaScript object notation, JSON) remote procedure call (JSON remote procedure call, JSON-RPC).

[00338] The IDE frontend may notify the language service of an operation performed by the user. For example, the IDE frontend may send a code operation performed by the user to the language service in a form of a request. The language service may process the request, and return a processing result to the IDE frontend in a form of a response.

[00339] In step 630, the first instance is an instance that supports the first programming language. The first IDE backend in this instance is configured to provide a language service of the programming language. Step 630 may also be understood as activating a language service of the first programming language.

[00340] The first operation may indicate the first programming language. For example, the first programming language may be determined based on a context of the first operation.

[00341] Step 630 may also be understood as activating a corresponding language service based on the context of the first operation.

[00342] The term "first" in "first operation" is merely intended to impose a limitation that the operation is related to activating the first IDE backend, and is not intended to impose any other limitation.

[00343] The following provides example descriptions of the first operation. The following provides example descriptions by using an example in which the first operation is an operation based on the first project.

[00344] For example, the first operation on the first project may be a creation operation on the first project.

[00345] Correspondingly, the first operation on the first project may be step 631.

[00346] When the user creates the first project, the system may activate a language service of a programming language related to the first operation. For example, the system may activate language services of all programming languages in the first project.

[00347] For example, the first operation on the first project may be an open operation on the first project.

[00348] Correspondingly, the first operation on the first project may be step 632.

[00349] For example, when the user opens the first project a first time, the system may activate

a language service of a programming language related to the first operation. For example, the system may activate language services of all programming languages in the first project.

[00350] In a possible implementation, the first operation may be an operation on a file in the first project.

[00351] For example, the first operation may be an open operation on a first file in the first project.

[00352] For example, when the user opens the first file a first time, the system may activate a language service of a programming language related to the first operation. That the user opens the first file may be understood as opening the first file in response to a file opening request of the user, where the file opening request indicates to open the first file.

[00353] The first file may be a code file, or may be a non-code file.

[00354] The term "first" in "first file" is merely for ease of description, and is not intended to impose any other limitation.

[00355] For example, the first file is a code file. For example, the first file is a first code file, the first operation may be an open operation on the first code file, and the first programming language may be a programming language of the first code file. When the user opens the first code file a first time, a language service of a programming language used by the first code file may be activated. For example, the first code file is a Java file. When the user opens the Java file, a language service of Java may be activated.

[00356] For example, the first file is a non-code file. When the user opens the first file a first time, language services of all programming languages in the first project may be activated. For example, the first file may be a 1^{st} file opened in the first project. To be specific, when the user opens the 1^{st} file in the first project, the language services of all the programming languages in the first project may be activated.

[00357] Activating the corresponding language service based on the first operation may be immediately activating the corresponding language service in response to the first operation. Alternatively, activating the corresponding language service based on the first operation may be activating the corresponding language service after a period of time in response to the first operation. This is not limited in this embodiment of this application.

[00358] It should be understood that the foregoing is merely an example of the first operation, and specific content of the first operation is not limited in this embodiment of this application.

[00359] Activating the first IDE backend in the instance may include starting the first IDE backend in the instance.

[00360] In a process of activating the first IDE backend in the first instance, whether the first IDE backend is deployed in the first instance may be first checked.

[00361] For example, the IDE frontend may check whether the first IDE backend is deployed in the first instance.

[00362] Optionally, activating the first IDE backend in the first instance may include: starting the first IDE backend when the first IDE backend is deployed in the first instance.

[00363] Optionally, activating the first IDE backend in the first instance may include: when the first IDE backend is not deployed in the first instance, deploying the first IDE backend in the first instance, and starting the first IDE backend.

[00364] Further, the method 600 may further include: closing the first IDE backend based on a second operation.

[00365] In other words, stopping the first IDE backend service based on the second operation may also be understood as stopping a language service of a programming language related to the second operation.

[00366] The term "second" in "second operation" is merely used to impose a limitation that the second operation is related to closing the first IDE backend, and is not intended to impose any other limitation.

[00367] A specific first IDE backend to be closed may be set according to a requirement. For example, a to-be-closed first IDE backend is related to the second operation.

[00368] The following provides example descriptions of the second operation.

[00369] For example, the second operation may be performed on the first project.

[00370] For example, the second operation on the first project may include a close operation on the first project.

[00371] When the user closes the first project, the system may close the first IDE backend. For example, the system may close all started first IDE backends. That the user closes the first project may be understood as closing the first project in response to a project closing request of the user, where the project closing request indicates to close the first project.

[00372] For example, the second operation may be performed on the IDE frontend.

[00373] For example, the second operation on the IDE frontend may include a close operation on the IDE frontend.

[00374] When the user closes the IDE frontend, the system may close the first IDE backend. For example, the system may close all started first IDE backends. That the user closes the IDE frontend may be understood as closing the IDE frontend in response to an IDE closing request of the user, where the IDE closing request indicates to close the IDE frontend.

[00375] Closing the first IDE backend based on the second operation may be: immediately closing the first IDE backend in response to the second operation. Alternatively, closing the first IDE backend based on the second operation may be: closing the first IDE backend after a period

of time in response to the second operation. This is not limited in this embodiment of this application.

[00376] It should be understood that the foregoing is merely an example of the second operation, and specific content of the second operation is not limited in this embodiment of this application.

[00377] In addition, the method 600 may be applied in a remote development scenario. In other words, the first IDE backend is deployed in a remote environment.

[00378] In this case, before starting the first IDE backend in the first instance, the IDE frontend may establish a connection to the first instance. For example, the IDE frontend may establish a secure shell protocol (secure shell protocol, SSH) connection to the first instance.

[00379] In this case, activating the first IDE backend in the instance may further include: establishing a persistent connection between the IDE frontend and the first IDE backend, so that the first IDE backend can be in a standby state at any time, thereby improving interaction efficiency.

[00380] For example, after the first IDE backend in the first instance is started, a remote port monitored by the first IDE backend may be sent to the IDE frontend. The IDE frontend may establish an SSH tunnel between a local port and the remote port (local port:remote port), and upgrade a hypertext transfer protocol (hypertext transfer protocol, HTTP) connection to a WebSocket connection. A subsequent service request may be exchanged through this connection.

[00381] In addition, the SSH tunnel may be further closed when the first IDE backend is closed.

[00382] In step 630, after the corresponding language service is activated, the corresponding language service may perform corresponding processing. Specifically, the code operation of the user, for example, editing, debugging, testing, compilation, and building, may be sent, in a form of a request, by the IDE frontend to the corresponding language service for processing. The language service may return a processing result to the IDE frontend, and the IDE frontend displays the processing result to the user.

[00383] Optionally, the method 600 may include: The IDE frontend sends a first request to the first IDE backend in the first environment; the first IDE backend in the first environment generates a first response to the first request; and the first IDE backend in the first environment returns the first response to the IDE frontend.

[00384] In this case, the IDE frontend cooperates with the first IDE backend in the first environment, to implement software development in a system in which the first IDE backend is located. That is, a specific operation of software development may be completed by the first IDE backend. The first request may be generated based on the code operation of the user.

[00385] Optionally, the IDE may further include a second IDE backend running in a second environment. In this case, the method 600 may further include:

[00386] The IDE frontend sends a second request to the first IDE backend in the first

environment.

[00387] The first IDE backend in the first environment forwards the second request to the second IDE backend in the second environment.

[00388] The second IDE backend in the second environment generates a second response to the second request.

[00389] The second IDE backend in the second environment sends the second response to the first IDE backend in the first environment.

[00390] The first IDE backend in the first environment returns the second response to the IDE frontend.

[00391] The IDE frontend, the first IDE backend, and the second IDE backend cooperate to implement software development in a system in which the second IDE backend is located. That is, a specific operation of software development may be completed by the second IDE backend. The second request may be generated based on the code operation of the user.

[00392] For related descriptions of the second IDE backend, refer to the foregoing descriptions. Details are not described herein again.

[00393] Further, the method 600 may further include: The first IDE backend may project, via a display server protocol, a graphical user interface program running in a remote environment onto a local system in which the IDE frontend is located.

[00394] For descriptions of application projection, refer to the foregoing descriptions. Details are not described herein again.

[00395] Further, the method 600 may further include step 640.

[00396] 640: Destroy the one or more instances.

[00397] Optionally, step 640 may include: destroying the one or more instances based on a third operation.

[00398] The term "third" in "third operation" is merely used to impose a limitation that the third operation is related to instance destruction, and is not intended to impose any other limitation.

[00399] The following provides example descriptions of the third operation.

[00400] For example, the third operation may be performed on the IDE frontend.

[00401] For example, the third operation on the IDE frontend may include a close operation on the IDE frontend.

[00402] When the user closes the IDE frontend, the system may destroy the one or more instances. For example, the system may destroy all instances related to the IDE frontend.

[00403] The third operation and the second operation may be a same operation, or may be different operations. For example, the third operation and the second operation may be close operations on the IDE frontend. To be specific, when the user closes the IDE frontend, the system

may close the first IDE backend, and destroy all the instances related to the IDE frontend. For another example, the third operation may be a close operation on the IDE frontend, and the second operation may be a close operation on the first project. To be specific, when the user closes the first project, the system may close the first IDE backend; and when the user closes the IDE frontend, the system may destroy all the instances related to the IDE frontend.

[00404] It should be understood that the foregoing is merely an example of the third operation, and specific content of the third operation is not limited in this embodiment of this application.

[00405] Destroying the one or more instances based on the third operation may be: immediately destroying the one or more instances in response to the third operation. Alternatively, destroying the one or more instances based on the third operation may be: destroying the one or more instances after a period of time in response to the third operation. This is not limited in this embodiment of this application.

[00406] FIG. 10 to FIG. 17 are diagrams of applying a solution to different development scenarios according to an embodiment of this application. IDE architectures in FIG. 10 to FIG. 17 may be the IDE architecture shown in FIG. 2. For detailed descriptions of the IDE architecture, refer to FIG. 2. To avoid repetition, a part of descriptions are properly omitted when FIG. 10 to FIG. 17 are described. In solutions shown in FIG. 10 to FIG. 17, a local system may also be understood as a local operating system.

[00407] For ease of description, in the solutions in FIG. 10 to FIG. 17, an IDE frontend is represented as a frontend UI. A user may interact with the frontend UI.

[00408] FIG. 10 is a diagram of applying a solution to a local IDE development scenario according to an embodiment of this application. In the local IDE development scenario, an IDE frontend and an IDE backend (for example, a first IDE backend in FIG. 10) run in a same local operating system. The local IDE development scenario may also be replaced with a desktop IDE development scenario.

[00409] In a development scenario involving a local IDE, a frontend UI runs in a client program of the local IDE, and the client program of the local IDE is deployed in a local system. The client program of the local IDE may also be replaced with a client program of a desktop IDE.

[00410] Specifically, in the development scenario shown in FIG. 10, the first IDE backend is also deployed in the local system. A first environment in which the first IDE backend is located is the local system in which the IDE frontend is located. In other words, the IDE frontend and the IDE backend run in a same local operating system.

[00411] In this scenario, communication between the IDE frontend and the first IDE backend may be implemented through inter-process communication. For example, as shown in FIG. 10, the IDE frontend and the first IDE backend may interact with each other through inter-process

communication (inter-process communication, IPC).

[00412] FIG. 11 is a diagram of applying a solution to a web IDE development scenario according to an embodiment of this application. In the development scenario shown in FIG. 11, a user may access a backend server through a local browser, to implement software development. An IDE backend (for example, a first IDE backend shown in FIG. 11) is deployed on the backend server. Alternatively, it may be understood that the backend server is configured to provide an IDE backend service, and may also be referred to as an IDE backend server. The IDE backend may also be referred to as an IDE backend of a web IDE.

[00413] In the development scenario involving the web IDE, an IDE frontend runs in a client program of the browser, as shown in FIG. 11. The client program of the browser is deployed in a local system. The local browser is the client program, of the browser, deployed in the local system.

[00414] An environment in which the first IDE backend is located may be a local backend server, or may be a remote backend server. In other words, the first IDE backend may be deployed in the local backend server, or may be deployed in the remote backend server. Correspondingly, the user may access the IDE backend through the local browser, to implement software development locally or remotely.

[00415] For ease of description, in the solution in this embodiment of this application, the local backend server and the remote backend server each may be considered as a remote environment.

[00416] As shown in FIG. 11, a system to which a first environment belongs may be a homogeneous system, or may be a heterogeneous system. In other words, the first IDE backend may be deployed in the homogeneous system, or may be deployed in the heterogeneous system. In other words, the user may access the IDE backend through the local browser, to implement software development in the homogeneous system or the heterogeneous system. Implementing software development in the homogeneous system or the heterogeneous system may be replaced with developing an application in the homogeneous system or the heterogeneous system.

[00417] The first IDE backend may be deployed on a plurality of types of compute resources. For example, the first IDE backend may be deployed on a container or a virtual machine. For example, a compute resource of the remote environment may include any one or more of the following: a local container, a local virtual machine, a remote container, or a remote virtual machine. The first IDE backend may be deployed in any one of the foregoing.

[00418] In the development scenario shown in FIG. 11, communication between the IDE frontend and the first IDE backend may be implemented through communication between the browser and the backend server. For example, as shown in FIG. 11, the IDE frontend and the first IDE backend may interact with each other based on an HTTP.

[00419] FIG. 12 is a diagram of applying a solution to another web IDE development scenario

according to an embodiment of this application. A main difference between FIG. 11 and FIG. 12 lies in that a first IDE backend in FIG. 12 runs in a client program of a browser. In other words, a host process (host process) of the first IDE backend may be a browser process, and an IDE frontend and the first IDE backend may run in different browser processes.

[00420] In the development scenario shown in FIG. 12, communication between the IDE frontend and the first IDE backend may be implemented through inter-process communication. For example, as shown in FIG. 12, the IDE frontend and the first IDE backend may interact with each other through IPC.

[00421] FIG. 13 is a diagram of applying a solution to a development scenario of remote development of a local IDE according to an embodiment of this application. A user may access a backend server through a client program of the local IDE, to implement software development. An IDE backend (for example, a first IDE backend in FIG. 13) is deployed on the backend server.

[00422] In the development scenario involving the local IDE, a frontend UI runs in a client program of the local IDE, and the client program of the local IDE is deployed in a local system.

[00423] A first environment may be a local backend server, or may be a remote backend server. In other words, the first IDE backend may be deployed in the local backend server, or may be deployed in the remote backend server. Correspondingly, the user may access the IDE backend through the client program of the local IDE, to implement software development locally or remotely.

[00424] As shown in FIG. 13, a system to which the first environment belongs may be a homogeneous system and/or a heterogeneous system. In other words, the first IDE backend may be deployed in the homogeneous system, or may be deployed in the heterogeneous system. In other words, the user may access the IDE backend through the client program of the local IDE, to implement software development in the homogeneous system or the heterogeneous system.

[00425] The first IDE backend may be deployed on a plurality of types of compute resources. For example, the first IDE backend may be deployed on a container or a virtual machine. For example, a compute resource of the remote environment may include any one or more of the following: a local container, a local virtual machine, a remote container, or a remote virtual machine. The first IDE backend may be deployed in any one of the foregoing.

[00426] In the development scenario shown in FIG. 13, communication between the IDE frontend and the first IDE backend may be implemented through communication between a client and a server. For example, as shown in FIG. 13, the IDE frontend and the first IDE backend may interact with each other through an SSH tunnel, and the IDE frontend and a corresponding language service may communicate with each other through JSON RPC. For detailed descriptions, refer to the method 600 shown in FIG. 9.

[00427] FIG. 14 is a diagram of applying a solution to a development scenario of remote development of a web IDE according to an embodiment of this application. In the development scenario involving the web IDE, a frontend UI runs in a client program of a browser, and the client program of the browser is deployed in a local system.

[00428] In the development scenario shown in FIG. 14, a user may access a first IDE backend in a remote environment #1 (an example of a first environment) through a local browser, and interact with a second IDE backend in a remote environment #2 (an example of a second environment) through the first IDE backend in the remote environment #1, to perform software development in the remote environment #2. The first IDE backend in the remote environment #1 is an IDE backend of the web IDE. The remote environment #1 and the remote environment #2 are different remote environments. To be specific, the user may develop, through the IDE frontend of the local browser, an application in another remote environment different from a remote environment in which the IDE backend of the web IDE is located.

[00429] As shown in FIG. 14, the IDE backend may be deployed in different remote environments.

[00430] A system to which the first environment belongs may be a homogeneous system and/or a heterogeneous system. In other words, the first IDE backend may be deployed in the homogeneous system, or may be deployed in the heterogeneous system. A system to which the second environment belongs may be a homogeneous system and/or a heterogeneous system. In other words, the second IDE backend may be deployed in the homogeneous system, or may be deployed in the heterogeneous system. In other words, the user may implement software development in the homogeneous system or the heterogeneous system of the remote environment #2 through the IDE frontend of the local browser.

[00431] The IDE backend may be deployed on a plurality of types of compute resources. For detailed descriptions, refer to the foregoing descriptions. Details are not described herein again.

[00432] The following provides example descriptions of the development scenario shown in FIG. 14.

[00433] For example, the user may log in to a cloud account through the local browser. The IDE frontend running in the local browser uses the cloud account to be connected to a compute service on the cloud server, and creates a service instance. A hardware architecture type preconfigured by the user may be a type #1. In this case, a service instance is created on hardware of the type #1, an IDE backend (an example of the first IDE backend) on the service instance is started, and software development in the remote environment #1 (an example of the first environment) is performed. The IDE backend is the IDE backend in the remote environment #1. In a development process, the user may need to perform software development on hardware of another hardware architecture

type (for example, a type #2). In this case, in the solution in this embodiment of this application, the IDE backend in the remote environment #1 may establish a connection to the remote environment #2 (an example of the second environment), and hardware of the remote environment #2 includes hardware of the type #2. A service instance is created on the hardware of the type #2, an IDE backend (an example of the second IDE backend) on the service instance is started, thereby implementing software development in the remote environment #2.

[00434] In the development scenario shown in FIG. 14, communication between the IDE frontend and the IDE backend of the web IDE may be implemented through communication between a browser and a server, and communication between the IDE backend of the web IDE and an IDE backend in another remote environment may be implemented through communication between servers.

[00435] For example, as shown in FIG. 14, in a software development process, the IDE frontend and the IDE backend of the web IDE may communicate with each other based on an HTTP, and the IDE backend of the web IDE and the IDE backend in another remote environment may interact through an SSH tunnel, for example, interact through JSON RPC.

[00436] As described in the foregoing method 600, code activities of the user, such as editing, debugging, testing, compilation, and building, may be sent, in a form of a request, by the IDE frontend to a corresponding language service for processing. The language service may return a processing result to the IDE frontend, and the IDE frontend displays the processing result to the user. In the solution shown in FIG. 14, the language service is provided by the IDE backend (the second IDE backend in FIG. 14) in the remote environment #2. Specifically, the IDE frontend may send the request of the code activity of the user to the IDE backend (the first IDE backend in FIG. 14) of the web IDE by using an HTTP request, and the IDE backend of the web IDE forwards the request to the IDE backend in the remote environment #2 through an SSH tunnel established between the IDE backend of the web IDE and the remote environment #2, to process the request in an IDE backend component in the remote environment, so that the user can develop an application in the homogeneous system or the heterogeneous system in the remote environment #2 on the IDE frontend of the local browser.

[00437] FIG. 15 to FIG. 17 are diagrams of three types of application projection according to an embodiment of this application. In FIG. 15 to FIG. 17, a GUI program A to a GUI program D in a local system are a GUI program A to a GUI program D that are to be projected.

[00438] FIG. 15 is a diagram of applying a solution to a local IDE development scenario according to an embodiment of this application. A main difference between the solution in FIG. 15 and the solution in FIG. 10 lies in that the solution shown in FIG. 15 relates to application projection. To avoid repetition, a difference between the solution shown in FIG. 15 and the solution

shown in FIG. 10 is mainly described when the solution shown in FIG. 15 is described. For other descriptions, refer to the solution in FIG. 10.

[00439] A first IDE backend may establish a connection to one or more remote environments, so that a GUI program in the one or more remote environments can be projected onto the local system.

[00440] For example, the first IDE backend may separately establish an SSH tunnel with the one or more remote environments, so that the GUI program in the one or more remote environments can be projected onto the local system.

[00441] There may be one or more GUI programs in the one or more remote environments. For example, as shown in FIG. 15, the first IDE backend respectively establishes SSH tunnels with two remote environments, so that four GUI programs (namely, a GUI program A, a GUI program B, a GUI program C, and a GUI program D) in the two remote environments can be projected onto the local system, and the four GUI programs can be simultaneously displayed in a frontend UI.

[00442] It should be understood that a quantity of remote environments in which GUI programs are located and a quantity of GUI programs in each remote environment in FIG. 15 are merely examples, and constitute no limitation on the solution in this embodiment of this application.

[00443] FIG. 16 is a diagram of applying a solution to a web IDE development scenario according to an embodiment of this application. A main difference between the solution in FIG. 16 and the solution in FIG. 11 lies in that the solution shown in FIG. 16 relates to application projection. To avoid repetition, a difference between the solution shown in FIG. 16 and the solution shown in FIG. 11 is mainly described when the solution shown in FIG. 16 is described. For other descriptions, refer to the solution in FIG. 11.

[00444] In FIG. 16, a first IDE backend is deployed in a first environment. In the solution shown in FIG. 16, the first environment is different from a third environment. The first IDE backend may establish a connection to one or more third environments, so that a GUI program in the one or more third environments can be projected onto a local system.

[00445] For example, the first IDE backend may establish an SSH tunnel with the one or more third environments, so that a GUI program in the one or more third environments can be projected onto the local system.

[00446] As described above, a user may access, through a local browser, the first IDE backend deployed in the first environment. For example, the local browser may establish an SSH tunnel with the first environment, namely, an SSH tunnel browser (SSH-in-browser), and the first environment establishes an SSH tunnel with the third environment, so that the GUI program in the third environment can be projected onto the local system.

[00447] There may be one or more GUI programs in the one or more third environments. For

example, as shown in FIG. 16, the user may access, through the local browser, first IDE backends deployed in two first environments. The local browser respectively establishes SSH tunnels with the two first environments, and the two first environments respectively establish SSH tunnels with two third environments. Four GUI programs in the two third environments may be projected onto the local system via a display server protocol, and the four GUI programs may be simultaneously displayed in a frontend UI.

[00448] It should be understood that a quantity of remote environments in which GUI programs are located and a quantity of GUI programs in each remote environment in FIG. 16 are merely examples, and constitute no limitation on the solution in this embodiment of this application.

[00449] FIG. 17 is a diagram of applying a solution to a development scenario of remote development of a local IDE according to an embodiment of this application. A main difference between the solution in FIG. 17 and the solution in FIG. 13 lies in that the solution shown in FIG. 17 relates to application projection. To avoid repetition, a difference between the solution shown in FIG. 17 and the solution shown in FIG. 13 is mainly described when the solution shown in FIG. 17 is described. For other descriptions, refer to the solution in FIG. 13.

[00450] A first IDE backend is deployed in a first environment. In the solution shown in FIG. 17, the first environment is different from a third environment. In FIG. 17, the first IDE backend may establish a connection to one or more third environments, so that a GUI program in the one or more third environments can be projected onto a local system.

[00451] For example, the first IDE backend may establish an SSH tunnel with the one or more third environments, so that a GUI program in the one or more third environments can be projected onto the local system.

[00452] The user may access a backend server in the first environment through a client program of a local IDE, to implement software development. For example, the client program of the local IDE may establish an SSH tunnel with the first environment, and the first environment establishes an SSH tunnel with the third environment, so that the GUI program in the third environment can be projected onto the local system.

[00453] There may be one or more GUI programs in the one or more third environments. For example, as shown in FIG. 17, the user may access, through the client program of the local IDE, first IDE backends deployed in two first environments. The client program of the local IDE respectively establishes SSH tunnels with the two first environments, and the two first environments respectively establish SSH tunnels with two third environments. Four GUI programs in the two third environments may be projected onto the local system via a display server protocol, and the four GUI programs may be simultaneously displayed in a frontend UI.

[00454] It should be understood that a quantity of remote environments in which GUI programs

are located and a quantity of GUI programs in each remote environment in FIG. 17 are merely examples, and constitute no limitation on the solution in this embodiment of this application.

[00455] FIG. 18 is a flowchart of a software development method according to an embodiment of this application. The method shown in FIG. 18 may be considered as a specific implementation of the method 600 shown in FIG. 9. For detailed descriptions, refer to the method 600. To avoid repetition, a part of descriptions are properly omitted when the method 700 is described. An IDE in the method 700 uses the IDE architecture shown in FIG. 2. To be specific, an IDE backend service in the method 700 may be the first IDE backend 220 in FIG. 2 and/or the first IDE backend in FIG. 6, and an IDE frontend in the method 700 may be the IDE frontend 210 in FIG. 2. A service instance in the method 700 is the instance in the method 600.

[00456] The method 700 shown in FIG. 18 may be applied in a development scenario of remote development of a local IDE, for example, the development scenario shown in FIG. 13.

[00457] For example, the method 700 shown in FIG. 18 may be applied in a scenario in which a local device is combined with a cloud server. In other words, the method 700 may be applied in a scenario in which a local device is combined with a cloud compute service. In other words, the first IDE backend 220 may be deployed on the cloud compute service.

[00458] As shown in FIG. 18, the method 700 may include the following steps.

[00459] 710: Start an IDE.

[00460] A user starts a client program of a local IDE, in other words, starts an IDE frontend, in other words, starts the IDE.

[00461] The IDE frontend may be automatically connected to the cloud compute service by using a configured account, to create the service instance.

[00462] For example, after the user starts the IDE, a login interface may be displayed, to prompt the user to perform account login. After the user logs in, the IDE frontend may be connected to the cloud compute service by using an account of the user, and create service instances of various programming languages for the user. For example, as shown in FIG. 18, a service instance of Java and a service instance of Python are created for the user.

[00463] It should be understood that, that the service instance in the solution shown in FIG. 18 is implemented by using a container is merely an example, and constitutes no limitation on the solution in this embodiment of this application. For example, in another implementation, the service instance may be implemented by using a virtual machine.

[00464] It should be understood that a quantity of service instances and a corresponding programming language in the solution shown in FIG. 18 are merely examples, and constitute no limitation on the solution in this embodiment of this application. In another implementation, another quantity of service instances and/or a service instance of another programming language

may be further created.

[00465] A hardware architecture type of the service instance may be preconfigured by the user. For example, if the user preconfigures that the hardware architecture type of the service instance is a hardware type #1, a service instance is created for the user based on a hardware architecture of the hardware type #1.

[00466] 720: Create a project.

[00467] For example, the user may create the project by using a project template provided by the IDE.

[00468] A multi-language project is used as an example. The user may create the multi-language project by using the project template provided by the IDE. For example, programming languages involved in the multi-language project are Java and Python.

[00469] 730: Obtain an activation operation (an example of a first operation).

[00470] When the IDE frontend obtains an activation operation of the user, a corresponding language service may be activated.

[00471] When the user performs some operations, the IDE may automatically activate the corresponding language service based on a current context, that is, activate a context-aware language service. These operations are activation operations.

[00472] For example, the activation operation may be opening a code file in a project, namely, a language file. For example, when the user opens a Java file a first time, the IDE may activate a language service of Java based on a current context. For another example, when the user opens a Python file a first time, the IDE may activate a language service of Python based on a current context.

[00473] The following provides example descriptions of an activation process by using an example of activating the language service of Java. For example, after the user creates a multi-language project including Java and Python, an activation process of a language service of Java can be triggered by opening a Java file in the project. The activation process may include the following steps.

[00474] 731: The IDE frontend establishes an SSH connection to the service instance of Java.

[00475] 732: Start an IDE backend service in the service instance of Java, and send a remote port monitored by the IDE backend service to the IDE frontend.

[00476] Specifically, the IDE frontend checks whether the IDE backend service is deployed in a container (namely, a Java container) in the service instance. If the IDE backend service is deployed in the container in the service instance, the IDE backend service is started. If the IDE backend service is not deployed in the container in the service instance, the IDE backend service is deployed in the container, and the IDE backend service is started.

[00477] The remote port monitored by the IDE backend service is sent to the IDE frontend.

[00478] 733: The IDE frontend establishes an SSH tunnel between a local port and the remote port (local port:remote port), and upgrades an HTTP connection to a WebSocket connection. In this way, all subsequent service requests may be exchanged through this connection.

[00479] If the user does not need the IDE backend service currently, the IDE backend service may be stopped. A triggering manner of stopping the IDE backend service may be set according to a requirement.

[00480] For example, if a project is closed, an IDE backend service related to the project may be stopped. For example, after creating the multi-language project including Java and Python, the user performs the activation operation, so that the language service of Java and the language service of Python are activated, in other words, an IDE backend service used to provide the language service of Java and the language service of Python is started. If the user closes the language project, the IDE backend service used to provide the language service of Java and the language service of Python may be stopped.

[00481] For example, if the IDE frontend is closed, all IDE backend services interacting with the IDE frontend may be stopped.

[00482] Further, the SSH tunnel may be closed.

[00483] 740: Obtain a code operation.

[00484] When the IDE frontend obtains a code operation of the user, the code operation of the user may be sent to the language service in a form of a service request for processing. After processing the code operation, the language service returns a processing result to the IDE frontend. The IDE frontend displays the result to the user.

[00485] 750: Close the IDE.

[00486] If the user closes the IDE, the IDE frontend may notify the compute service to destroy a service instance.

[00487] For example, when the user closes the IDE, the IDE frontend may notify the compute service to destroy all service instances created when the IDE frontend is started.

[00488] The solution shown in FIG. 18 is used as an example. When the user closes the IDE, two service instances in FIG. 18 may be destroyed.

[00489] The following describes an apparatus in embodiments of this application with reference to FIG. 19 to FIG. 22. It should be understood that the following described apparatus can perform the method in embodiments of this application. To avoid unnecessary repetition, repeated descriptions are properly omitted when the apparatus in embodiments of this application is described below.

[00490] FIG. 19 is a block diagram of a software development system according to an

embodiment of this application. A system 2000 shown in FIG. 19 may be configured to perform the method shown in FIG. 9 and/or FIG. 18.

[00491] As shown in FIG. 19, the system 2000 may include an IDE frontend 2010 and a first IDE backend 2020. The IDE frontend may be the IDE frontend 210 in FIG. 2. The first IDE backend 2020 may be the first IDE backend 220 in FIG. 2.

[00492] The first IDE backend 2020 supports running in a plurality of types of environments, and the plurality of types of environments include at least one of the following: a plurality of types of operating systems or a plurality of types of hardware architectures.

[00493] The IDE frontend 2010 is configured to: obtain user configuration information, where the configuration information includes at least one of the following: a hardware architecture type or an operating system type; and deploy the first IDE backend 2020 in a first environment based on the configuration information, so that software development on the first IDE backend 2020 in the first environment is implemented by using the IDE frontend 2010, where the first environment is an environment determined based on the configuration information in the plurality of types of environments supported by the first IDE backend.

[00494] Optionally, the IDE frontend 2010 runs in a client program of an IDE or a client program of a browser.

[00495] Optionally, the IDE frontend 2010 includes a user interface UI component and an adaptation layer, the UI component is configured to provide a user interface, and the adaptation layer is configured to provide a unified interface for the UI component, so that the UI component runs in one or more client programs.

[00496] Optionally, the first IDE backend 2020 includes an IDE kernel, and the IDE kernel is configured to provide language features of a plurality of programming languages.

[00497] Optionally, the DE kernel includes: an indexing system, configured to store index information of source code; a language parsing adaptation layer, configured to invoke a language parser corresponding to a programming language of the source code to parse the source code; and a language feature layer, configured to provide language features of a plurality of programming languages, where the plurality of programming languages include the programming language of the source code.

[00498] Optionally, the plurality of types of environments include a local system in which the IDE frontend 2010 is located and/or another environment different from the local system.

[00499] Optionally, the first environment includes one or more instances, and the IDE frontend 2010 is specifically configured to: obtain the one or more instances based on the configuration information, where the first IDE backend 2020 is deployed in the one or more instances; or obtain the one or more instances based on the configuration information; and when the first IDE backend

2020 is not deployed in a part or all of the one or more instances, deploy the first IDE backend 2020 in the instance in which the first IDE backend 2020 is not deployed.

[00500] Optionally, the IDE frontend 2010 is further configured to start the first IDE backend 2020 in a first instance in the one or more instances, so that the first IDE backend 2020 in the first instance provides a language feature of a first programming language supported by the first instance.

[00501] Optionally, the IDE frontend 2010 is further configured to establish a persistent connection to the first IDE backend 2020 in the first instance.

[00502] Optionally, the IDE frontend 2010 is specifically configured to start the first IDE backend 2020 in the first instance based on a first operation. The first operation indicates the first programming language.

[00503] Optionally, the first operation includes an open operation of a first code file, and the first programming language is a programming language of the first code file.

[00504] Optionally, the configuration information is indicated by a login request carrying verification information.

[00505] Optionally, the IDE frontend 2010 is further configured to send a first request to the first IDE backend 2020 in the first environment; and the first IDE backend 2020 in the first environment is configured to: generate a first response to the first request; and return the first response to the IDE frontend 2010.

[00506] Optionally, the system further includes a second IDE backend running in the second environment, and the IDE frontend 2010 is further configured to send a second request to the first IDE backend 2020 in the first environment; the first IDE backend 2020 in the first environment is configured to forward the second request to the second IDE backend in the second environment; the second IDE backend in the second environment is configured to: generate a second response to the second request; and send the second response to the first IDE backend 2020 in the first environment; and the first IDE backend 2020 in the first environment is further configured to return the second response to the IDE frontend 2010.

[00507] Optionally, the first IDE backend 2020 in the first environment is further configured to project, via a display server protocol, a graphical user interface program running in a remote environment onto the local system in which the IDE frontend 2010 is located.

[00508] For detailed descriptions, refer to the method 600 and the method 700. Details are not described herein again.

[00509] Each module in the system 2000 may be implemented by using software or hardware. For example, the following uses the first IDE backend 2020 as an example to describe an implementation of the first IDE backend 2020. Similarly, for an implementation of another module,

refer to an implementation of the first IDE backend 2020.

[00510] A module is used as an example of a software functional unit, and the first IDE backend 2020 may include code that runs on a computing instance. The computing instance may include at least one of a physical host (compute device), a virtual machine, and a container. Further, there may be one or more computing instances. For example, the first IDE backend 2020 may include code that runs on a plurality of hosts/virtual machines/containers. It should be noted that the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same region (region), or may be distributed in different regions. Further, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same availability zone (availability zone, AZ), or may be distributed in different AZs. Each AZ includes one data center or a plurality of data centers that are geographically close to each other. Usually, one region may include a plurality of AZs.

[00511] Similarly, the plurality of hosts/virtual machines/containers configured to run the code may be distributed in a same virtual private cloud (virtual private cloud, VPC), or may be distributed in a plurality of VPCs. Usually, one VPC is disposed in one region. A communication gateway needs to be disposed in each VPC for communication between two VPCs in a same region and cross-region communication between VPCs in different regions. The VPCs are interconnected through the communication gateway.

[00512] A module is used as an example of a hardware functional unit, and the first IDE backend 2020 may include at least one compute device, for example, a server. Alternatively, the first IDE backend 2020 may be a device implemented by using an application-specific integrated circuit (application-specific integrated circuit, ASIC) or a programmable logic device (programmable logic device, PLD), or the like. The PLD may be implemented by a complex programmable logic device (complex programmable logic device, CPLD), a field-programmable gate array (field-programmable gate array, FPGA), generic array logic (generic array logic, GAL), or any combination thereof.

[00513] A plurality of compute devices included in the first IDE backend 2020 may be distributed in a same region, or may be distributed in different regions. The plurality of compute devices included in the first IDE backend 2020 may be distributed in a same AZ, or may be distributed in different AZs. Similarly, the plurality of compute devices included in the first IDE backend 2020 may be distributed in a same VPC, or may be distributed in a plurality of VPCs. The plurality of compute devices may be any combination of compute devices such as a server, an ASIC, a PLD, a CPLD, an FPGA, and GAL.

[00514] It should be noted that, in another embodiment, the first IDE backend 2020 may be configured to perform any step in the software development method, other modules may be

configured to implement any step in the software development method, steps that each module is responsible for implementing may be specified according to a requirement, and all modules separately implement different steps in the software development method to implement all functions of the system 2000.

[00515] This application further provides a compute device 1000. As shown in FIG. 20, the compute device 1000 includes a bus 1002, a processor 1004, a memory 1006, and a communication interface 1008. The processor 1004, the memory 1006, and the communication interface 1008 communicate with each other through the bus 1002. The compute device 1000 may be a server or a terminal device. It should be understood that a quantity of processors and a quantity of memories in the compute device 1000 are not limited in this application.

[00516] The bus 1002 may be a peripheral component interconnect (peripheral component interconnect, PCI) bus, an extended industry standard architecture (extended industry standard architecture, EISA) bus, or the like. Buses may be classified into an address bus, a data bus, a control bus, and the like. For ease of representation, only one line is used in FIG. 20 for representation, but it does not mean that there is only one bus or one type of bus. The bus 1002 may include a path for information transfer between various components (such as the memory 1006, the processor 1004, and the communication interface 1008) in the compute device 1000.

[00517] The processor 1004 may include any one or more of processors such as a central processing unit (central processing unit, CPU), a graphics processing unit (graphics processing unit, GPU), a microprocessor (microprocessor, MP), or a digital signal processor (digital signal processor, DSP).

[00518] The memory 1006 may include a volatile memory (volatile memory), for example, a random access memory (random access memory, RAM). The processor 1004 may further include a non-volatile memory (non-volatile memory), for example, a read-only memory (read-only memory, ROM), a flash memory, a hard disk drive (hard disk drive, HDD), or a solid-state drive (solid-state drive, SSD).

[00519] The memory 1006 stores executable program code, and the processor 1004 executes the executable program code to separately implement functions of the foregoing modules, to implement the software development method. In other words, the memory 1006 stores instructions used to perform the software development method.

[00520] The communication interface 1008 uses a transceiver module, for example but not limited to, a network interface card or a transceiver, to implement communication between the compute device 1000 and another device or a communication network.

[00521] An embodiment of this application further provides a compute device cluster. The compute device cluster includes at least one compute device. The compute device may be a server,

for example, a central server, an edge server, or a local server in a local data center. In some embodiments, the compute device may alternatively be a terminal device, for example, a desktop computer, a notebook computer, or a smartphone.

[00522] As shown in FIG. 21, the compute device cluster includes at least one compute device 1000. A memory 1006 in one or more compute devices 1000 in the compute device cluster may store same instructions used to perform the software development method.

[00523] In some possible implementations, the memory 1006 in the one or more compute devices 1000 in the compute device cluster may also separately store some instructions used to perform the software development method. In other words, a combination of the one or more compute devices 1000 may jointly execute the instructions used to perform the software development method.

[00524] It should be noted that memories 1006 in different compute devices 1000 in the compute device cluster may store different instructions respectively used to perform some functions of the software development system. In other words, the instructions stored in the memories 1006 in the different compute devices 1000 may implement functions of one or more modules in the system 2000.

[00525] In some possible implementations, the one or more compute devices in the compute device cluster may be connected through a network. The network may be a wide area network, a local area network, or the like.

[00526] FIG. 22 shows a possible implementation. As shown in FIG. 22, two compute devices 1000A and 1000B are connected through a network. Specifically, each compute device is connected to the network through a communication interface in the compute device. In this possible implementation, a memory 1006 of the compute device 1000A stores instructions for performing functions of an IDE frontend 2010. In addition, a memory 1006 in the compute device 1000B stores instructions for performing functions of a first IDE backend 2020.

[00527] It should be understood that functions of the compute device 1000A shown in FIG. 22 may also be completed by a plurality of compute devices 1000. Similarly, functions of the compute device 1000B may also be completed by a plurality of compute devices 1000.

[00528] An embodiment of this application further provides a computer program product including instructions. The computer program product may be software or a program product that includes instructions and that can run on a compute device or can be stored in any usable medium. When the computer program product runs on at least one compute device, the at least one compute device is enabled to perform the software development method.

[00529] An embodiment of this application further provides a computer-readable storage medium. The computer-readable storage medium may be any usable medium that can be stored

by a compute device, or a data storage device, such as a data center, including one or more usable media. The usable medium may be a magnetic medium (for example, a floppy disk, a hard disk, or a magnetic tape), an optical medium (for example, a DVD), a semiconductor medium (for example, a solid-state drive), or the like. The computer-readable storage medium includes instructions, and the instructions instruct a compute device to perform the software development method.

[00530] A person of ordinary skill in the art may be aware that, in combination with the examples described in embodiments disclosed in this specification, units and algorithm steps may be implemented by 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 this application.

[00531] 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.

[00532] In the several embodiments provided in this application, it should be understood that the disclosed system, apparatus, and method may be implemented in another manner. For example, the described apparatus embodiments are merely examples. For example, division into the units is merely logical functional division and may be other division in actual implementation. For example, a plurality of units or components may be combined or integrated into another system, or some features may be ignored or not performed. In addition, the displayed or discussed mutual couplings or direct couplings or communication connections may be implemented through some interfaces. The indirect couplings or communication connections between the apparatuses or units may be implemented in electronic, mechanical, or other forms.

[00533] The units described as separate parts may or may not be physically separate, and parts displayed as units may or may not be physical units, may be located in one position, or may be distributed on a plurality of network units. Some or all of the units may be selected based on an actual requirement, to achieve the objectives of the solutions of embodiments.

[00534] In addition, functional units in embodiments of this application may be integrated into one processing unit, each of the units may exist alone physically, or two or more units are integrated into one unit.

[00535] When the functions are implemented in the form of a software functional unit and sold or used as an independent product, the functions may be stored in a computer-readable storage

medium. Based on such an understanding, the technical solutions of this application essentially, or the part contributing to the conventional technologies, 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 a plurality of instructions for instructing a computer device (which may be a personal computer, a server, or a network device) to perform all or some of the steps of the processing 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.

[00536] The foregoing descriptions are merely specific implementations of this application, but are not intended to limit the protection scope 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 of this application. Therefore, the protection scope of this application shall be subject to the protection scope of the claims.

CLAIMS

1. A software development method, applied in an integrated development environment, IDE, wherein the IDE comprises an IDE frontend and a first IDE backend, the first IDE backend supports running in a plurality of types of environments, the plurality of types of environments comprise at least one of the following: a plurality of types of operating systems or a plurality of types of hardware architectures, and the method comprises:

obtaining, by the IDE frontend, user configuration information, wherein the configuration information comprises at least one of the following: a hardware architecture type or an operating system type; and

deploying, by the IDE frontend, the first IDE backend in a first environment based on the configuration information, so that software development on the first IDE backend in the first environment is implemented by using the IDE frontend, wherein the first environment is an environment determined based on the user configuration information in the plurality of types of environments supported by the first IDE backend.

2. The method according to claim 1, wherein the IDE frontend runs in a client program of the IDE or a client program of a browser.

3. The method according to claim 2, wherein the IDE frontend comprises a user interface UI component and an adaptation layer, the UI component is configured to provide a user interface, and the adaptation layer is configured to provide a unified interface for the UI component, so that the UI component runs in one or more client programs.

4. The method according to any one of claims 1 to 3, wherein the first IDE backend comprises an IDE kernel, and the IDE kernel is configured to provide language features of a plurality of programming languages.

5. The method according to claim 4, wherein the IDE kernel comprises:

an indexing system, configured to store index information of source code;

a language parsing adaptation layer, configured to invoke a language parser corresponding to a programming language of the source code to parse the source code; and

a language feature layer, configured to provide the language features of the plurality of programming languages, wherein the plurality of programming languages comprise the programming language of the source code.

6. The method according to any one of claims 1 to 5, wherein the plurality of types of environments comprise a local system in which the IDE frontend is located and/or another environment different from the local system.

7. The method according to any one of claims 1 to 6, wherein the first environment comprises

one or more instances, and the deploying, by the IDE frontend, the first IDE backend in the first environment based on the configuration information comprises:

obtaining, by the IDE frontend, the one or more instances based on the configuration information, wherein the first IDE backend is deployed in the one or more instances; or

obtaining, by the IDE frontend, the one or more instances based on the configuration information; and

when the first IDE backend is not deployed in a part or all of the one or more instances, deploying, by the IDE frontend, the first IDE backend in the instance in which the first IDE backend is not deployed.

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

starting, by the IDE frontend, the first IDE backend in a first instance in the one or more instances, so that the first IDE backend in the first instance provides a language feature of a first programming language supported by the first instance.

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

establishing, by the IDE frontend, a persistent connection to the first IDE backend in the first instance.

10. The method according to claim 8 or 9, wherein the starting, by the IDE frontend, the first IDE backend in the first instance in the one or more instances comprises:

starting, by the IDE frontend, the first IDE backend in the first instance based on a first operation, wherein the first operation indicates the first programming language.

11. The method according to claim 10, wherein the first operation comprises an open operation of a first code file, and the first programming language is a programming language of the first code file.

12. The method according to any one of claims 1 to 11, wherein the configuration information is indicated by a login request carrying verification information.

13. The method according to any one of claims 1 to 12, wherein the method further comprises:

sending, by the IDE frontend, a first request to the first IDE backend in the first environment;

generating, by the first IDE backend in the first environment, a first response to the first request; and

returning, by the first IDE backend in the first environment, the first response to the IDE frontend.

14. The method according to any one of claims 1 to 12, wherein the IDE further comprises a second IDE backend running in a second environment, and the method further comprises:

sending, by the IDE frontend, a second request to the first IDE backend in the first environment;

forwarding, by the first IDE backend in the first environment, the second request to the second IDE backend in the second environment;

generating, by the second IDE backend in the second environment, a second response to the second request;

sending, by the second IDE backend in the second environment, the second response to the first IDE backend in the first environment; and

returning, by the first IDE backend in the first environment, the second response to the IDE frontend.

15. The method according to any one of claims 1 to 14, wherein the method further comprises:

projecting, by the first IDE backend in the first environment via a display server protocol, a graphical user interface program running in a remote environment onto the local system in which the IDE frontend is located.

16. A software development system, comprising an integrated development environment, IDE frontend and a first IDE backend, wherein the first IDE backend supports running in a plurality of types of environments, and the plurality of types of environments comprise at least one of the following: a plurality of types of operating systems or a plurality of types of hardware architectures; and

the IDE frontend is configured to:

obtain user configuration information, wherein the configuration information comprises at least one of the following: a hardware architecture type or an operating system type; and

deploy the first IDE backend in a first environment based on the configuration information, so that software development on the first IDE backend in the first environment is implemented by using the IDE frontend, wherein the first environment is an environment determined based on the user configuration information in the plurality of types of environments supported by the first IDE backend.

17. The system according to claim 16, wherein the IDE frontend runs in a client program of an IDE or a client program of a browser.

18. The system according to claim 17, wherein the IDE frontend comprises a user interface UI component and an adaptation layer, the UI component is configured to provide a user interface, and the adaptation layer is configured to provide a unified interface for the UI component, so that the UI component runs in one or more client programs.

19. The system according to any one of claims 16 to 18, wherein the first IDE backend comprises an IDE kernel, and the IDE kernel is configured to provide language features of a plurality of programming languages.

20. A computer program product comprising instructions, wherein when the instructions are

run by a compute device cluster, the compute device cluster is enabled to perform the method according to any one of claims 1 to 15.