CLAIMS

What is claimed is:

1. A method for executing a smart contract, comprising:

receiving, by a blockchain node, a request for creating a smart contract, wherein the request comprises code of the smart contract, and the code of the smart contract comprises hotspot code marked by using a hotspot identifier;

after deploying the smart contract, starting, by the blockchain node, to compile the code of the smart contract into machine code through Just-In-Time (JIT) compilation, and storing machine code compiled from the hotspot code marked by using the hotspot identifier; and

executing, by the blockchain node when executing the deployed smart contract, the stored machine code if the machine code corresponding to the code of the smart contract is locally stored.

2. The method according to claim 1, wherein

the code of the smart contract is bytecode generated through compilation of code written in a high-level language; or

the method further comprises: when the code of the smart contract is written in a high-level language, compiling, by the blockchain node through a compiler, the code written in the high-level language to generate bytecode.

3. The method according to claim 2, wherein the code of the smart contract is written by a user, and a writing location comprises one of: a local device of the user, the blockchain node, or another blockchain node different from the blockchain node.

4. The method according to claim 2, wherein the code in the form of high-level language comprises hotspot code marked by using a first hotspot identifier, bytecode generated by compiling the code written in the high-level language comprises hotspot code marked by using a second hotspot identifier, and the first hotspot identifier corresponds to the second hotspot identifier.

5. The method according to claim 1, wherein the starting, by the blockchain node, to compile the code of the smart contract into machine code through Just-In-Time (JIT) compilation comprises:

compiling, by the blockchain node, the code of the smart contract through JIT compilation after creating the smart contract; or

compiling, by the blockchain node, the code of the smart contract through JIT compilation in a relatively idle time period after creating the smart contract; or

compiling, by the blockchain node, the code of the smart contract through JIT compilation after receiving a transaction request for invoking the created smart contract.

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

identifying non-marked hotspot code other than the hotspot code marked by using the hotspot identifier, and compiling and storing the non-marked hotspot code, in a process of translating and/or executing the code of the smart contract.

7. The method according to claim 6, wherein the identifying non-marked hotspot code other than the hotspot code marked by using the hotspot identifier, and compiling and storing the non-marked hotspot code, in a process of translating and/or executing the code of the smart contract comprises:

if a size of storage space for storing the machine code exceeds a size of space occupied by the hotspot code marked by using the hotspot identifier, identifying the non-marked hotspot code other than the hotspot code marked by using the hotspot identifier, and compiling and storing the non-marked hotspot code, in the process of translating and/or executing the code of the smart contract.

8. The method according to claim 6, wherein a granularity of identifying the non-marked hotspot code comprises a function level or a code block level.

9. The method according to claim 6, wherein the compiling the code of the smart contract into machine code through JIT compilation comprises:

performing optimized compilation in a process of translating and/or executing the hotspot code marked by using the hotspot identifier and/or the non-marked hotspot code.

10. The method according to claim 9, wherein performing the optimized compilation on the code comprises at least one of:

compiling with reference to contextual relevance of the code, wherein the contextual relevance comprises hardware information and environment information of the blockchain node, and context information of the code;

eliminating dead code through data flow analysis or live variable analysis;

optimizing loop statements in the code; and

replacing common word expressions in the code.

11. The method according to claim 1, wherein the executing, by the blockchain node, the machine code comprises:

executing, by the blockchain node, stored machine code corresponding to a function to be invoked or a code block in the smart contract.

12. The method according to claim 1, wherein when executing the deployed smart contract, the blockchain node interprets and executes the code of the smart contract if the machine code of the smart contract is not locally stored.

13. The method according to claim 12, wherein the interpreting and executing, by the blockchain node, the code of the smart contract comprises:

interpreting and executing, by the blockchain node, code corresponding to a function to be invoked in the smart contract.

14. The method according to claim 12, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the code of the smart contract if the machine code of the smart contract is not locally stored comprises:

when the blockchain node executes the deployed smart contract, if the machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current JIT compilation status of the code of the smart contract, and interpreting and executing the code of the smart contract if the code of the smart contract is being compiled through JIT compilation; or

when the blockchain node executes the deployed smart contract, if the machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current JIT compilation status of the code of the smart contract, and

if JIT compilation has not been started, starting JIT compilation, and interpreting and executing the code of the smart contract.

15. The method according to claim 1, wherein the memory is a cache.

16. The method according to claim 1, wherein executing the smart contract is performed by a virtual machine, and the virtual machine comprises one of: a Java Virtual Machine (JVM), a WebAssembly (WASM) virtual machine, a Dalvik virtual machine, or an Android Runtime (ART) virtual machine.

17. The method according to claim 1, wherein the blockchain node belongs to a public blockchain, a private blockchain, or a consortium blockchain.

18. A method for executing a smart contract, wherein code of the smart contract comprises hotspot code marked by using a hotspot identifier, and the method comprises:

after deploying the smart contract, starting , by a blockchain node, to compile the code of the smart contract into machine code through JIT compilation after deploying the code of the smart contract, and storing machine code compiled from the hotspot code marked by using the hotspot identifier; and

executing, by the blockchain node when executing the deployed smart contract, the stored machine code if the machine code corresponding to the code of the smart contract is locally stored.

19. The method according to claim 18, wherein

the code of the smart contract is bytecode generated through compilation of code written in high-level language; or

the method further comprises: when the code of the smart contract is written in a high-level language, compiling, by the blockchain node through a compiler, the code written in the high-level language to generate bytecode.

20. The method according to claim 19, wherein the code of the smart contract is written by a user, and a writing location comprises one of: a local device of the user, the blockchain node, or another blockchain node different from the blockchain node.

21. The method according to claim 19, wherein the code in the form of high-level language comprises hotspot code marked by using a first hotspot identifier, bytecode generated by compiling the code written in the high-level language comprises hotspot code marked by using a second hotspot identifier, and the first hotspot identifier corresponds to the second hotspot identifier.

22. The method according to claim 18, wherein the starting, by the blockchain node, to compile the code of the smart contract into machine code through Just-In-Time (JIT) compilation comprises:

compiling, by the blockchain node, the code of the smart contract through JIT compilation after creating the smart contract; or

compiling, by the blockchain node, the code of the smart contract through JIT compilation in a relatively idle time period after creating the smart contract; or

compiling, by the blockchain node, the code of the smart contract through JIT compilation after receiving a transaction request for invoking the created smart contract.

23. The method according to claim 18, further comprising:

identifying non-marked hotspot code other than the hotspot code marked by using the hotspot identifier, and compiling and storing the non-marked hotspot code, in a process of translating and/or executing the code of the smart contract.

24. The method according to claim 23, wherein the identifying non-marked hotspot code other than the hotspot code marked by using the hotspot identifier, and compiling and storing the non-marked hotspot code, in a process of translating and/or executing the code of the smart contract comprises:

if a size of storage space for storing the machine code exceeds a size of space occupied by the hotspot code marked by using the hotspot identifier, identifying the non-marked hotspot code other than the hotspot code marked by using the hotspot identifier, and compiling and storing the non-marked hotspot code, in the process of translating and/or executing the code of the smart contract.

25. The method according to claim 23, wherein a granularity of identifying the non-marked hotspot code comprises a function level or a code block level.

26. The method according to claim 23, wherein the compiling the code of the smart contract into machine code through JIT comprises:

performing optimized compilation in a process of translating and/or executing the hotspot code marked by using the hotspot identifier and/or the non-marked hotspot code.

27. The method according to claim 26, wherein performing the optimized compilation on the code comprises at least one of:

compiling with reference to contextual relevance of the code, wherein the contextual relevance comprises hardware information and environment information of the blockchain node, and context information of the code;

eliminating dead code through data flow analysis or live variable analysis;

optimizing loop statements in the code; and

replacing common subexpressions in the code.

28. The method according to claim 18, wherein the executing, by the blockchain node, the machine code comprises:

executing, by the blockchain node, stored machine code corresponding to a function to be invoked or a code block in the smart contract.

29. The method according to claim 18, wherein when executing the deployed smart contract, the blockchain node interprets and executes the code of the smart contract if the machine code of the smart contract is not locally stored.

30. The method according to claim 29, wherein the interpreting and executing, by the blockchain node, the code of the smart contract comprises:

interpreting and executing, by the blockchain node, code corresponding to a function to be invoked in the smart contract.

31. The method according to claim 29, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the code of the smart contract if the machine code of the smart contract is not locally stored comprises:

when the blockchain node executes the deployed smart contract, if the machine code of the smart contract is not locally stored, further querying, by the

blockchain node, a current JIT compilation status of the code of the smart contract, and interpreting and executing the code of the smart contract if the code of the smart contract is being compiled through JIT compilation; or

when the blockchain node executes the deployed smart contract, if the machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current JIT compilation status of the code of the smart contract, and if JIT compilation has not been started, starting JIT compilation, and interpreting and executing the code of the smart contract.

32. The method according to claim 18, wherein the memory is a cache.

33. The method according to claim 18, wherein the virtual machine comprises one of: a Java Virtual Machine (JVM), a WebAssembly (WASM) virtual machine, a Dalvik virtual machine, or an Android Runtime (ART) virtual machine.

34. The method according to claim 18, wherein the blockchain node belongs to a public blockchain, a private blockchain, or a consortium blockchain.

35. A blockchain node for executing a smart contract, comprising:

a processor; and

a memory, wherein the memory stores a program, and when the processor executes the program, the method according to any one of claims 1 to 17 or 18 to 34 is performed.

36. A storage medium, configured to store a program, wherein when the program is executed, the operation according to any one of claims 1 to 17 or 18 to 34 is performed.