CLAIMS

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

after deploying bytecodes of the smart contract;

respectively starting, by a first blockchain node and a second blockchain node, to compile the bytecodes of the smart contract into machine codes through a Just-In-Time (JIT) Compiler and store the machine codes;

when executing the deployed smart contract, executing, by the first blockchain node, the machine codes if the machine codes of the smart contract are locally stored;

determining, by the second blockchain node, whether execution results of the machine codes and the bytecodes are consistent by:

comparing test results of the machine codes and the bytecodes in at least one verification scenario, wherein consistency of the test results indicates that the execution results of the machine codes and the bytecodes are consistent; and/or

performing formal verification for the machine codes and the bytecodes, wherein consistency of verification results indicates that the execution results of the machine codes and the bytecodes are consistent; and

when executing the deployed smart contract, performing, by the second blockchain node, interpretation execution for the bytecodes in response to determining that the execution results of the machine codes and the bytecodes of the smart contract are consistent if the machine codes of the smart contract are not locally stored.

2. The method according to claim 1, wherein after deploying the bytecodes of the smart contract, respectively starting, by the first blockchain node and the second blockchain node, to compile the bytecodes of the smart contract into the machine codes through the JIT Compiler comprises:

after completing creation of the smart contract, respectively starting, by the first blockchain node and the second blockchain node, to perform JIT compilation for the bytecodes of the smart contract; or,

after completing creation of the smart contract, respectively performing, by the first blockchain node and the second blockchain node, JIT compilation for the bytecodes of the smart contract in a relatively idle time period; or,

performing, by the first blockchain node and the second blockchain node, JIT

compilation for the bytecodes of the smart contract respectively after receiving a transaction request for invoking the created smart contract.

3. The method according to claim 1, wherein compiling the bytecodes of the smart contract into the machine codes through the JIT Compiler and storing the machine codes comprises:

performing compilation and storage by identifying hotspot bytecodes in a process of interpreting and/or executing the bytecodes of the smart contract; and/or,

performing optimizing compilation and storage in a process of interpreting and/or executing the bytecodes of the smart contract.

4. The method according to claim 3, wherein a granularity of identifying the hotspot bytecodes comprises a function level or a code block level.

5. The method according to claim 1, wherein executing, by the first blockchain node, the machine codes comprises:

executing, by the first blockchain node, stored machine codes corresponding to an invoked function and/or code block in the smart contract.

6. The method according to claim 1, wherein performing, by the second blockchain node, the interpretation execution for the bytecodes of the smart contract comprises:

performing, by the second blockchain node, interpretation execution for bytecodes corresponding to an invoked function in the smart contract.

7. The method according to claim 6, wherein when executing the deployed smart contract, performing, by the second blockchain node, the interpretation execution for the bytecodes in response to determining that the execution results of the machine codes and the bytecodes of the smart contract are consistent if the machine codes of the smart contract are not locally stored comprises:

when the second blockchain node executes the deployed smart contract, if the machine codes of the smart contract are not locally stored, further querying, by the second blockchain node, current JIT compilation of the bytecodes of the smart contract, and if the JIT compilation is in progress, performing the interpretation execution for the bytecodes of the smart contract in response to determining that the execution results of the machine codes and the bytecodes of the smart contract are consistent; or,

when the second blockchain node executes the deployed smart contract, if the machine codes of the smart contract are not locally stored, further querying, by the second blockchain

node, current JIT compilation for the bytecodes of the smart contract, and if the JIT compilation is not started, starting the JIT compilation and performing the interpretation execution for the bytecodes of the smart contract in response to determining that the execution results of the machine codes and the bytecodes of the smart contract are consistent.

8. The method according to claim 1, wherein consistency of the execution results of the machine codes and the bytecodes comprises at least one of: consistency of returned contract execution results, consistency of formed contract storages, or consistency of consumed blockchain assets.

9. The method according to any one of the preceding claims, wherein storing is caching.

10. The method according to any one of the preceding claims, further comprising:

respectively receiving, by the first blockchain node and the second blockchain node, a request for creating the smart contract, wherein the request comprises the bytecodes of the smart contract.

11. A blockchain system for executing a smart contract, comprising a first blockchain node and a second blockchain node, wherein the first blockchain node is configured to perform operations performed by the first blockchain node in a method according to any one of the preceding claims, and the second blockchain node is configured to perform operations performed by the second blockchain node in a method according to any one of the preceding claims.

12. A method performed by a second blockchain node for executing a smart contract, comprising:

after deploying bytecodes of the smart contract, starting, by the second blockchain node, to compile the bytecodes of the smart contract into machine codes through a Just-In-Time (JIT) Compiler and store the machine codes;

determining, by the second blockchain node, whether execution results of the machine codes and the bytecodes are consistent by:

comparing test results of the machine codes and the bytecodes in at least one verification scenario, wherein consistency of the test results indicates that the execution results of the machine codes and the bytecodes are consistent; and/or

performing formal verification for the machine codes and the bytecodes, wherein consistency of verification results indicates that the execution results of the machine codes

and the bytecodes are consistent; and

when executing the deployed smart contract, performing, by the second blockchain node, interpretation execution for the bytecodes, in response to determining that the execution results of the machine codes and the bytecodes of the smart contract are consistent, if the machine codes of the smart contract are not locally stored.

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

a processor;

a memory storing programs thereon, wherein the processor is configured to execute the programs to perform the method according to claim 12.

14. Storage media for storing programs, wherein the programs, when executed by a processor, configure the processor to perform the method according to claim 12.

15. Storage media for storing programs, wherein the programs, when respectively executed by a first processor and a second processor, configure the first processor and the second processor to respectively perform a method according to claims 1 to 10.