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 a bytecode of the smart contract;

compiling, by the blockchain node, all of the bytecode of the smart contract into first machine code by using a global compilation method comprising Ahead of Time, AoT, compilation after deploying the smart contract, and storing the first machine code; and

compiling, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract by using a local compilation method to obtain second machine code if the first machine code of the smart contract is not locally stored, storing the second machine code, and executing the second machine code.

2. The method according to claim 1, wherein the compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract specifically comprises:comprises globally compiling all of the bytecode of the smart contract in an idle time period of the blockchain node after creating the smart contract; and

globallylocally compiling without waiting for completion of the global compilation method, by the blockchain node, the bytecode of when executing the smart contract after creating the smart contract; or

globally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contractby using a local compilation method comprising Just-In-Time, JIT, compilation to obtain second machine code if the first machine code of the smart contract is not locally stored and the bytecode of the smart contract is being globally compiled, storing the second machine code, and executing the second machine code.

~~3.~~ 2. The method according to claim 1, wherein the compiling the bytecode of the smart contract into first machine code by using a global compilation method comprises:

performing optimized compilation in a process of globally compiling the bytecode of the smart contract to obtain the first machine code.

~~4.~~ 3. The method according to claim 1 or 2, wherein the locally compiling, by the blockchain node when executing the ~~deployed~~ smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored specifically comprises:

when the blockchain node executes the ~~deployed~~ smart contract, if the first machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current global compilation status of the bytecode of the smart contract, and locally compiling the bytecode of the smart contract if the bytecode of the smart contract is being globally compiled; or

when the blockchain node executes the ~~deployed~~ smart contract, if the first machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current global compilation status of the bytecode of the smart contract, and if global compilation has not been started, starting global compilation, and locally compiling the bytecode of the smart contract.

~~5.~~ 4. The method according to claim 1 or ~~3~~2, wherein the locally compiling, by the blockchain node, the bytecode of the smart contract comprises:

locally compiling, by the blockchain node, bytecode corresponding to a function to be invoked in the smart contract.

~~6.~~ 5. The method according to claim 1 or ~~4~~3, wherein the locally compiling the bytecode of the smart contract to obtain second machine code, and storing the second machine code comprises:

identifying second machine code that serves as a hotspot in a process of translating and/or executing the bytecode of the smart contract, and storing the second machine code that serves as a hotspot for subsequent invocation; and/or

performing optimized compilation to obtain the second machine code in the process of translating and/or executing the bytecode of the smart contract, and storing the second machine code.

~~7.~~ 6. The method according to claim 1, wherein when executing the ~~deployed~~ smart contract, the blockchain node executes the first machine code if the first machine code of the smart contract is locally stored.

~~8.~~ 7. The method according to claim ~~7~~6, wherein the executing, by the blockchain node, the first machine code comprises:

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

8.

~~9.~~The method according to any one of the preceding claims ~~1, 3, 7,~~ further comprising:

interpreting and ~~8, wherein the memory is a cache.~~

~~10. The method according to any one of claims 1 to 4, 7, and 8, wherein the global compilation method comprises Ahead of Time (AoT) or Just In Time (JIT), and the local compilation method comprises JIT.~~

~~11. A method for~~ executing ~~a smart contract, comprising:~~

~~compiling, by a blockchain node, bytecode of a smart contract into first machine code by using a global compilation method after deploying the smart contract, and storing the first machine code; and~~

~~compiling~~, by the blockchain node when executing the ~~deployed~~ smart contract, the bytecode of the smart contract ~~by using a local compilation method to obtain second machine code~~ if the first machine code of the smart contract is not locally stored~~, storing the second machine code, and executing the second machine code~~.

~~12.~~

9. The method according to claim ~~11, wherein the compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract specifically comprises:~~

~~globally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or~~

~~globally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract.~~

~~13. The method according to claim 11, wherein the compiling the bytecode of the smart contract into first machine code by using a global compilation method comprises:~~

~~performing optimized compilation in a process of globally compiling the bytecode of the smart contract to obtain the first machine code.~~

~~14. The method according to claim 11, wherein the locally compiling~~8, wherein the interpreting and executing, by the blockchain node when executing the ~~deployed~~ smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored specifically comprises:

when the blockchain node executes the ~~deployed smart contract, if the first machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current global compilation status of the bytecode of the smart contract, and locally compiling the bytecode of the smart contract if the bytecode of the smart contract is being globally compiled; or~~

~~when the blockchain node executes the deployed smart contract, if the first machine code of the smart contract is not locally stored, further querying, by the blockchain node, a current global compilation status of the bytecode of the smart contract, and if global compilation has not been started, starting global compilation, and locally compiling the bytecode of the smart contract.~~

~~15. The method according to claim 11 or 13, wherein the locally compiling, by the blockchain node, the bytecode of the smart contract comprises:~~

~~locally compiling, by the blockchain node, bytecode corresponding to a function to be invoked in the smart contract.~~

~~16. The method according to claim 11 or 14, wherein the locally compiling the bytecode of the smart contract to obtain second machine code, and storing the second machine code comprises:~~

~~identifying second machine code that serves as a hotspot in a process of translating and/or executing the bytecode of the smart contract, and storing the second machine code that serves as a hotspot for subsequent invocation; and/or~~

~~performing optimized compilation to obtain the second machine code in the process of translating and/or executing the bytecode of the smart contract, and storing the second machine code.~~

~~17. The method according to claim 11, wherein when executing the deployed smart contract, the blockchain node executes the first machine code if the first machine code of the smart contract is locally stored.~~

~~18. The method according to claim 17, wherein the executing, by the blockchain node, the first machine code comprises:~~

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

~~19. The method according to any one of claims 11, 13, 17, and 18, wherein the memory is a cache.~~

~~20. The method according to any one of claims 11 to 14, 17, and 18, wherein the global compilation method comprises AoT or JIT, and the local compilation method comprises JIT.~~

~~21. A blockchain node for executing a smart contract, configured to perform the method according to any one of claims 1 to 10 or 11 to 20.~~

~~22. 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 10 or 11 to 20 is performed.~~

~~23. A storage medium, configured to store a program, wherein when the program is executed, the operation according to any one of claims 1 to 10 or 11 to 20 is performed.~~

~~24. A method for executing a smart contract, comprising:~~

~~receiving, by a blockchain node, a request for creating a smart contract, wherein the request comprises bytecode of the smart contract;~~

~~compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract, and storing the first machine code; and~~

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

~~25. The method according to claim 24, wherein the compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract specifically comprises:~~

~~globally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or~~

~~globally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract.~~

~~26. The method according to claim 24, wherein the executing, by the blockchain node, the first machine code comprises:~~

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

~~27. The method according to claim 24, wherein when executing the deployed smart contract, the blockchain node interprets and executes the bytecode of the smart contract if the first machine code of the smart contract is not locally stored.~~

~~28. The method according to claim 27, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:~~

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

~~29. The method according to claim 27, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored specifically comprises:~~

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

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

~~30.~~ 10. The method according to ~~claim 24, wherein~~any one of the preceding claims, further comprising: executing, by the blockchain node when executing the ~~deployed~~ smart contract, the second machine code if the second machine code of the smart contract is locally stored.

11.~~blockchain node locally compiles and executes~~ The method according to any one of the preceding claims, further comprising: interpreting and executing, by the blockchain node when executing the smart contract, the bytecode of the smart contract if the ~~first~~second machine code of the smart contract is not locally stored.

~~31. The method according to claim 30, wherein the locally compiling the bytecode of the smart contract comprises:~~

~~translating the bytecode of the smart contract sentence by sentence, and caching translated machine code for subsequent invocation; or~~

identifying hotspot machine code in a process of translating and/or executing the bytecode of the smart contract, and compiling and caching the hotspot machine code for subsequent invocation.

32. The method according to claim 31, wherein a granularity of identifying the hotspot machine code comprises a function level or a code block level.

33. The method according to claim 31, wherein the locally compiling the bytecode of the smart contract further comprises: performing optimized compilation in a process of locally translating the bytecode of the smart contract.

34. The method according to any one of claims 24, 26, 27, 29, and 30, wherein the memory is a cache.

35. The method according to any one of claims 24 to 33, wherein the global compilation method comprises AoT or JIT, or the local compilation method comprises JIT.

36. 12. A method for executing a smart contract, comprising:

compiling, by a blockchain node, bytecode of a smart contract into first machine code by using a global compilation method after deploying the bytecode of the smart contract, and storing the first machine code; and

executing, by the blockchain node whenfor executing the deployeda smart contract, the first machine code if the first machine code of the smart contract is locally stored.comprising:

37. The method according to claim 36, wherein the compiling, by a blockchain node, bytecode of a smart contract into first machine code by using a global compilation method after deploying the bytecode of the smart contract specifically comprises:

globally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or

globally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract.

~~38. The method according to claim 37, wherein the executing, by the blockchain node, the first machine code comprises:~~

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

~~39. The method according to claim 37, wherein when executing the deployed smart contract, the blockchain node interprets and executes the bytecode of the smart contract if the first machine code of the smart contract is not locally stored.~~

~~40. The method according to claim 39, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:~~

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

~~41. The method according to claim 39, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored specifically comprises:~~

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

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

~~42. The method according to claim 36, wherein when executing the deployed smart contract, the blockchain node locally compiles and executes the bytecode of the smart contract if the first machine code of the smart contract is not locally stored.~~

~~43. The method according to claim 42, wherein the locally compiling the bytecode of the smart contract comprises:~~

~~translating the bytecode of the smart contract sentence by sentence, and caching translated machine code for subsequent invocation; or~~

~~identifying hotspot machine code in a process of translating and/or executing the bytecode of the smart contract, and compiling and caching the hotspot machine code for subsequent invocation.~~

~~44. The method according to claim 43, wherein a granularity of identifying the hotspot machine code comprises a function level or a code block level.~~

~~45. The method according to claim 42, wherein the locally compiling the bytecode of the smart contract further comprises: performing optimized compilation in a process of locally translating the bytecode of the smart contract.~~

~~46. The method according to any one of claims 36, 38, 39, 41, and 42, wherein the memory is a cache.~~

~~47. The method according to any one of claims 36 to 45, wherein the global compilation method comprises AoT or JIT, and the local compilation method comprises JIT.~~

~~48. A blockchain node for executing a smart contract, configured to perform the method according to any one of claims 24 to 35 or 36 to 47.~~

~~49. 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 the preceding claims ~~24 to 35 or 36 to 47~~ is

performed.

50. A storage medium, configured to store a program, wherein when the program is executed, the operation according to any one of claims 24 to 35 or 36 to 47 is performed.

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

receiving, by a blockchain node, a request for creating a smart contract, wherein the request comprises bytecode of the smart contract;

compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract, and storing the first machine code; and

interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored.

52. The method according to claim 51, wherein the compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract specifically comprises:

globally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or

globally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract.

53. The method according to claim 51, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:

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

54. The method according to claim 51, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored specifically comprises:

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

blockchain node, a current global compilation status of the bytecode of the smart contract, and interpreting and executing the bytecode of the smart contract if the bytecode of the smart contract is being globally compiled; or

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

55. The method according to claim 51, wherein when executing the deployed smart contract, the blockchain node executes the first machine code if the first machine code of the smart contract is locally stored.

56. The method according to claim 55, wherein the executing, by the blockchain node, the first machine code comprises:

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

57. The method according to any one of claims 51, 54, 55, and 56, wherein the memory is a cache.

58. The method according to any one of claims 51 to 56, wherein the global compilation method comprises AoT or JIT.

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

compiling, by a blockchain node, bytecode of a smart contract into first machine code by using a global compilation method after deploying the smart contract, and storing the first machine code; and

interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored.

~~60. The method according to claim 59, wherein the compiling, by the blockchain node, the bytecode of the smart contract into first machine code by using a global compilation method after deploying the smart contract specifically comprises:~~

~~globally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or~~

~~globally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract.~~

~~61. The method according to claim 59, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:~~

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

~~62. The method according to claim 59, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the first machine code of the smart contract is not locally stored specifically comprises:~~

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

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

~~63. The method according to claim 59, wherein when executing the deployed smart contract, the blockchain node executes the first machine code if the first machine code of the smart contract is locally stored.~~

64. The method according to claim 63, wherein the executing, by the blockchain node, the first machine code comprises:

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

65. The method according to any one of claims 59, 62, 63, and 64, wherein the memory is a cache.

66. The method according to any one of claims 59 to 64, wherein the global compilation method comprises AoT or JIT, and the local compilation method comprises JIT.

67. A blockchain node for executing a smart contract, configured to perform the method according to any one of claims 51 to 58 or 59 to 66.

68. 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 51 to 58 or 59 to 66 is performed.

69. A storage medium, configured to store a program, wherein when the program is executed, the operation according to any one of claims 51 to 58 or 59 to 66 is performed.

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

receiving, by a blockchain node, a request for creating a smart contract, wherein the request comprises bytecode of the smart contract;

compiling, by the blockchain node, the bytecode of the smart contract into second machine code by using a local compilation method after deploying the smart contract, and storing the second machine code; and

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

71. The method according to claim 70, wherein the compiling, by the blockchain node, the bytecode of the smart contract into second machine code by using a local compilation method after deploying the smart contract specifically comprises:

locally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or

locally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract; or

locally compiling, by the blockchain node, the bytecode of the smart contract after receiving a transaction request for invoking the created smart contract.

72. The method according to claim 70, wherein the compiling the bytecode of the smart contract into second machine code by using a local compilation method and storing the second machine code comprises:

identifying hotspot bytecode for compilation and storage in a process of translating and/or executing the bytecode of the smart contract; and/or

performing optimized compilation and storing optimized machine code in the process of translating and/or executing the bytecode of the smart contract.

73. The method according to claim 72, wherein a granularity of identifying the hotspot bytecode comprises a function level or a code block level.

74. The method according to claim 70, wherein the executing, by the blockchain node, the second machine code comprises:

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

75. The method according to claim 70, wherein when executing the deployed smart contract, the blockchain node interprets and executes the bytecode of the smart contract if the second machine code of the smart contract is not locally stored.

~~76. The method according to claim 75, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:~~

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

~~77. The method according to claim 75, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the second machine code of the smart contract is not locally stored specifically 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 local compilation status of the bytecode of the smart contract, and interpreting and executing the bytecode of the smart contract if the bytecode of the smart contract is being locally compiled; 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 local compilation status of the bytecode of the smart contract, and if local compilation has not been started, starting local compilation, and interpreting and executing the bytecode of the smart contract.~~

~~78. The method according to any one of claims 70, 72, 74, 75, and 77, wherein the memory is a cache.~~

~~79. The method according to any one of claims 70 to 77, wherein the local compilation method comprises JIT.~~

~~80. A method for executing a smart contract, comprising:~~

~~compiling, by a blockchain node, bytecode of a smart contract into second machine code by using a local compilation method after deploying the bytecode of the smart contract, and storing the second machine code; and~~

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

~~81. The method according to claim 80, wherein the compiling, by a blockchain node, bytecode of a smart contract into second machine code by using a local compilation method after deploying the bytecode of the smart contract specifically comprises:~~

~~locally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or~~

~~locally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract; or~~

~~locally compiling, by the blockchain node, the bytecode of the smart contract after receiving a transaction request for invoking the created smart contract.~~

~~82. The method according to claim 80, wherein the compiling bytecode of a smart contract into second machine code by using a local compilation method and storing the second machine code comprises:~~

~~identifying hotspot bytecode for compilation and storage in a process of translating and/or executing the bytecode of the smart contract; and/or~~

~~performing optimized compilation and storing optimized machine code in the process of translating and/or executing the bytecode of the smart contract.~~

~~83. The method according to claim 82, wherein a granularity of identifying the hotspot bytecode comprises a function level or a code block level.~~

~~84. The method according to claim 80, wherein the executing, by the blockchain node, the second machine code comprises:~~

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

~~85. The method according to claim 80, wherein when executing the deployed smart contract, the blockchain node interprets and executes the bytecode of the smart contract if the second machine code of the smart contract is not locally stored.~~

~~86. The method according to claim 85, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:~~

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

~~87. The method according to claim 86, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the second machine code of the smart contract is not locally stored specifically comprises:~~

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

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

~~88. The method according to any one of claims 80, 82, 84, 85, and 87, wherein the memory is a cache.~~

~~89. The method according to any one of claims 80 to 87, wherein the local compilation method comprises JIT.~~

~~90. A blockchain node for executing a smart contract, configured to perform the method according to any one of claims 70 to 79 or 80 to 89.~~

~~91. 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 70 to 79 or 80 to 89 is performed.~~

92. A storage medium, configured to store a program, wherein when the program is executed, the operation according to any one of claims 70 to 79 or 80 to 89 is performed.

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

receiving, by a blockchain node, a request for creating a smart contract, wherein the request comprises bytecode of the smart contract;

compiling, by the blockchain node, the bytecode of the smart contract into second machine code by using a local compilation method after deploying the smart contract, and storing the second machine code; and

interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the second machine code of the smart contract is not locally stored.

94. The method according to claim 93, wherein the compiling, by the blockchain node, the bytecode of the smart contract into second machine code by using a local compilation method after deploying the smart contract specifically comprises:

locally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or

locally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract; or

locally compiling, by the blockchain node, the bytecode of the smart contract after receiving a transaction request for invoking the created smart contract.

95. The method according to claim 93, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:

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

96. The method according to claim 93, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the second machine code of the smart contract is not locally stored specifically 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 local compilation status of the bytecode of the smart contract, and interpreting and executing the bytecode of the smart contract if the bytecode of the smart contract is being locally compiled; 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 local compilation status of the bytecode of the smart contract, and if local compilation has not been started, starting local compilation, and interpreting and executing the bytecode of the smart contract.~~

~~97. The method according to claim 93, wherein when executing the deployed smart contract, the blockchain node executes the second machine code if the second machine code of the smart contract is locally stored.~~

~~98. The method according to claim 97, wherein the executing, by the blockchain node, the second machine code comprises:~~

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

~~99. The method according to any one of claims 93, 96, 97, and 98, wherein the memory is a cache.~~

~~100. The method according to any one of claims 93 to 98, wherein the local compilation method comprises JIT.~~

~~101. A method for executing a smart contract, comprising:~~

~~compiling, by a blockchain node, bytecode of a smart contract into second machine code by using a local compilation method after deploying the smart contract, and storing the second machine code; and~~

~~interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the second machine code of the smart contract is not locally stored.~~

~~102. The method according to claim 101, wherein the compiling, by a blockchain node, bytecode of a smart contract into second machine code by using a local compilation method after deploying the smart contract specifically comprises:~~

~~locally compiling, by the blockchain node, the bytecode of the smart contract after creating the smart contract; or~~

~~locally compiling, by the blockchain node, the bytecode of the smart contract in an idle time period after creating the smart contract; or~~

~~locally compiling, by the blockchain node, the bytecode of the smart contract after receiving a transaction request for invoking the created smart contract.~~

~~103. The method according to claim 101, wherein the interpreting and executing, by the blockchain node, the bytecode of the smart contract comprises:~~

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

~~104. The method according to claim 101, wherein the interpreting and executing, by the blockchain node when executing the deployed smart contract, the bytecode of the smart contract if the second machine code of the smart contract is not locally stored specifically 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 local compilation status of the bytecode of the smart contract, and interpreting and executing the bytecode of the smart contract if the bytecode of the smart contract is being locally compiled; 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 local compilation status of the bytecode of the smart contract, and if local compilation has not been started, starting local compilation, and interpreting and executing the bytecode of the smart contract.~~

~~105. The method according to claim 101, wherein when executing the deployed smart contract, the blockchain node executes the second machine code if the second machine code of the smart contract is locally stored.~~

~~106. The method according to claim 105, wherein the executing, by the blockchain node, the second machine code comprises:~~

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

~~107. The method according to any one of claims 101, 104, 105, and 106, wherein the memory is a cache.~~

~~108. The method according to any one of claims 101 to 106, wherein the local compilation method comprises JIT.~~

~~109. A blockchain node for executing a smart contract, configured to perform the method according to any one of claims 93 to 100 or 101 to 108.~~

~~110. 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 93 to 100 or 101 to 108 is performed.~~

~~111. A storage medium, configured to store a program, wherein when the program is executed, the operation according to any one of claims 93 to 100 or 101 to 108 is performed.~~