Ethereum源码分析

什么是 Geth

1. Geth（Go-Ethereum） 是以太坊官方客户端（Ethereum Client）。
2. 运行后就能参与以太坊网络，维护区块链数据、验证交易和智能合约。

geth在以太坊中的作用

1. 网络入口

任何用户或开发者运行geth，就能成为以太坊网络中的一个节点。

geth负责：

维护区块链账本（区块和状态数据库）

验证区块、交易和智能合约

处理点对点网络通信

相当于以太坊网络的入口，让用户能直接与以太坊区块链交互，而不是依赖中心化的第三方(比如infura)

1. 执行层 （客户端）

以太坊在The Merge(合并)之后，分为两层：

执行层（Execution Layer, EL）：负责交易执行、EVM、账户状态维护（这就是 Geth 的职责）。

共识层（Consensus Layer, CL）：负责区块提议和验证（由 Prysm、Lighthouse 等客户端完成）。

在生态中，geth是目前最常用的执行层客户端

3、开发和测试工具

Geth 提供了丰富的命令行工具和 API，可以：

1、部署和测试智能合约

2、搭建私有链或测试链

3、查询账户余额、交易历史

4、使用控制台（JavaScript 控制台）和 JSON-RPC 与区块链交互

这让它成为开发者在本地实验和构建 DApp 的基础工具。

解析核心模块的交互关系

1. 区块链同步协议(eth/62, eth/63 → 现在是 snap/66 等)

定位：这是 Geth 节点之间 数据同步 的通信协议（基于 P2P 网络）。

功能：

节点启动时，通过 eth/62、eth/63 这样的协议与其他节点交换区块头、区块体、交易数据。

新版本还包括 snap 协议，用于高效状态同步。

交互关系：

同步下来的区块 → 提交到共识模块校验（比如 Ethash/PoS）；

校验通过后 → 交给 EVM 执行模块重放交易；

结果写入 区块链数据库（LevelDB/Freezer）；

同时，里面的未执行交易会被丢进 交易池。

1. 交易池管理和Gas机制

定位：交易池（TxPool）是本地缓存的“未打包交易池”。

功能：

用户通过 RPC 把交易发到节点 → 交易进入 TxPool；

交易根据 Gas Price / Priority Fee 排序；

节点会把交易广播给其他节点；

矿工/验证者在打包新区块时，从 TxPool 中挑选交易。

Gas 机制作用：

保证交易执行成本透明，防止恶意消耗资源；

通过竞价机制（Gas Price）让交易获得优先权。

交互关系：

TxPool 接收来自 RPC/API 的交易；

TxPool 也接收 P2P 网络传来的交易；

最终交易由 共识模块打包进区块；

区块里的交易会交给 EVM 模块执行。

3、EVM 执行环境构建

定位：以太坊虚拟机（EVM）是 Geth 的 核心执行引擎。

功能：

重放区块中的交易，修改链上状态；

负责合约部署、合约调用；

计算 Gas 消耗，确保执行资源有限。

交互关系：

输入：来自 交易池的交易，或来自 同步区块中的交易；

执行：EVM 解释字节码，修改 状态数据库（StateDB，Merkle Trie）；

输出：更新后的账户余额、合约存储、日志（事件）。

和 Gas 机制的关系：

每条 EVM 指令有固定 Gas 消耗；

EVM 在执行过程中实时扣除 Gas；

Gas 不够就触发 Out-of-Gas 错误 → 回滚状态，但 Gas 不退还。

4、共识算法实现(Ethash->PoS)

Ethash（工作量证明，PoW 时代）

矿工根据交易池中的交易构建候选区块；

执行 Ethash 算法（大量计算）寻找满足难度目标的哈希；

找到后广播新区块，其他节点验证该区块。

PoS（权益证明，The Merge 之后）

Geth 只负责 执行层，不再负责区块提议；

共识层（例如 Prysm、Lighthouse）通过 Engine API 与 Geth 通信；

共识层给 Geth 一个区块骨架，Geth 负责：

从 TxPool 挑交易填充区块；

执行交易（EVM）并生成状态根（StateRoot、ReceiptsRoot、LogsBloom）；

把执行结果返回给共识层，由 CL 最终完成区块确认。

整体交互流程图：

1、网络层（eth/62/63/snap）

下载新区块 → 验证区块头（共识模块）。

2、共识模块（Ethash/PoS）

判断区块是否合法 → 如果是新区块提议者，还要打包交易。

3、交易池（TxPool + Gas 机制）

提供候选交易，按 Gas 优先级排序。

4、EVM 执行模块

执行区块里的交易，修改状态树，扣除 Gas。

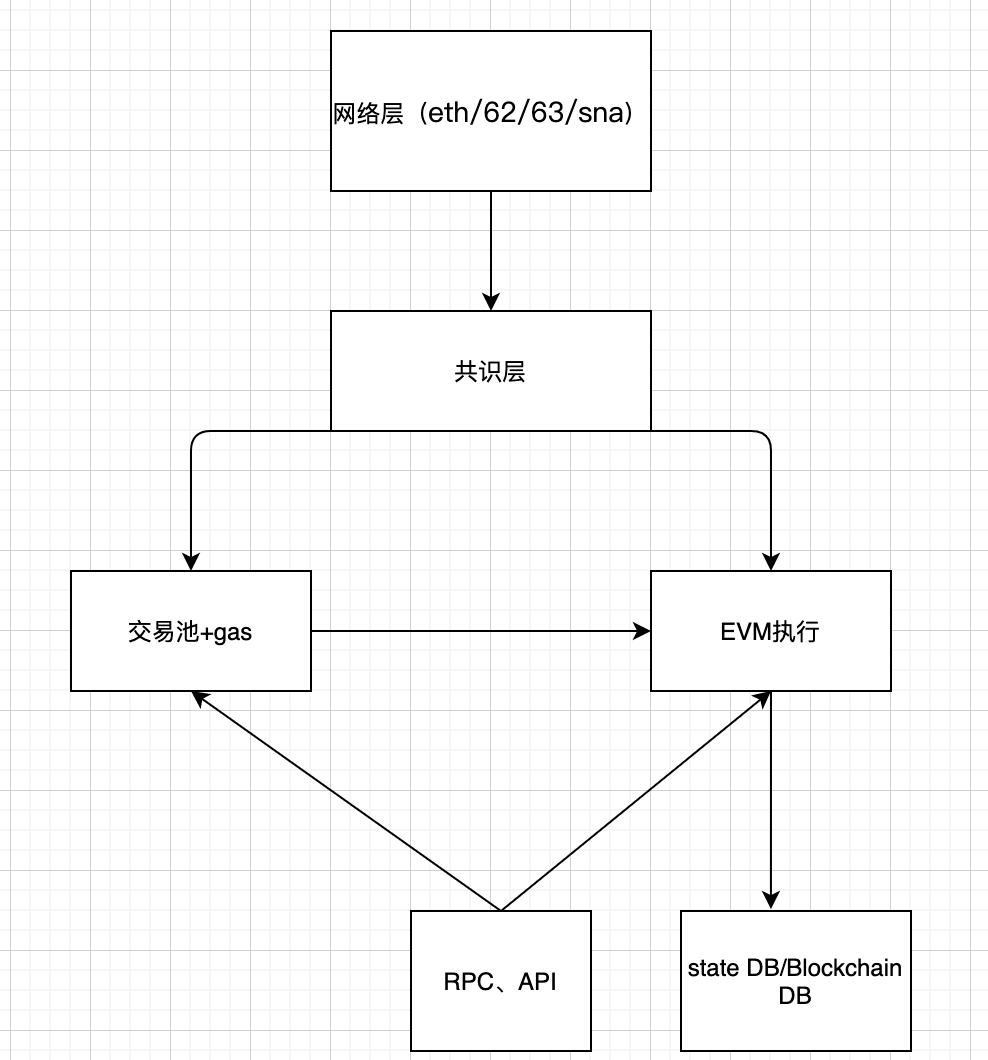
5、数据库（LevelDB/Freezer/Trie）

存储区块链数据、状态树、交易索引。

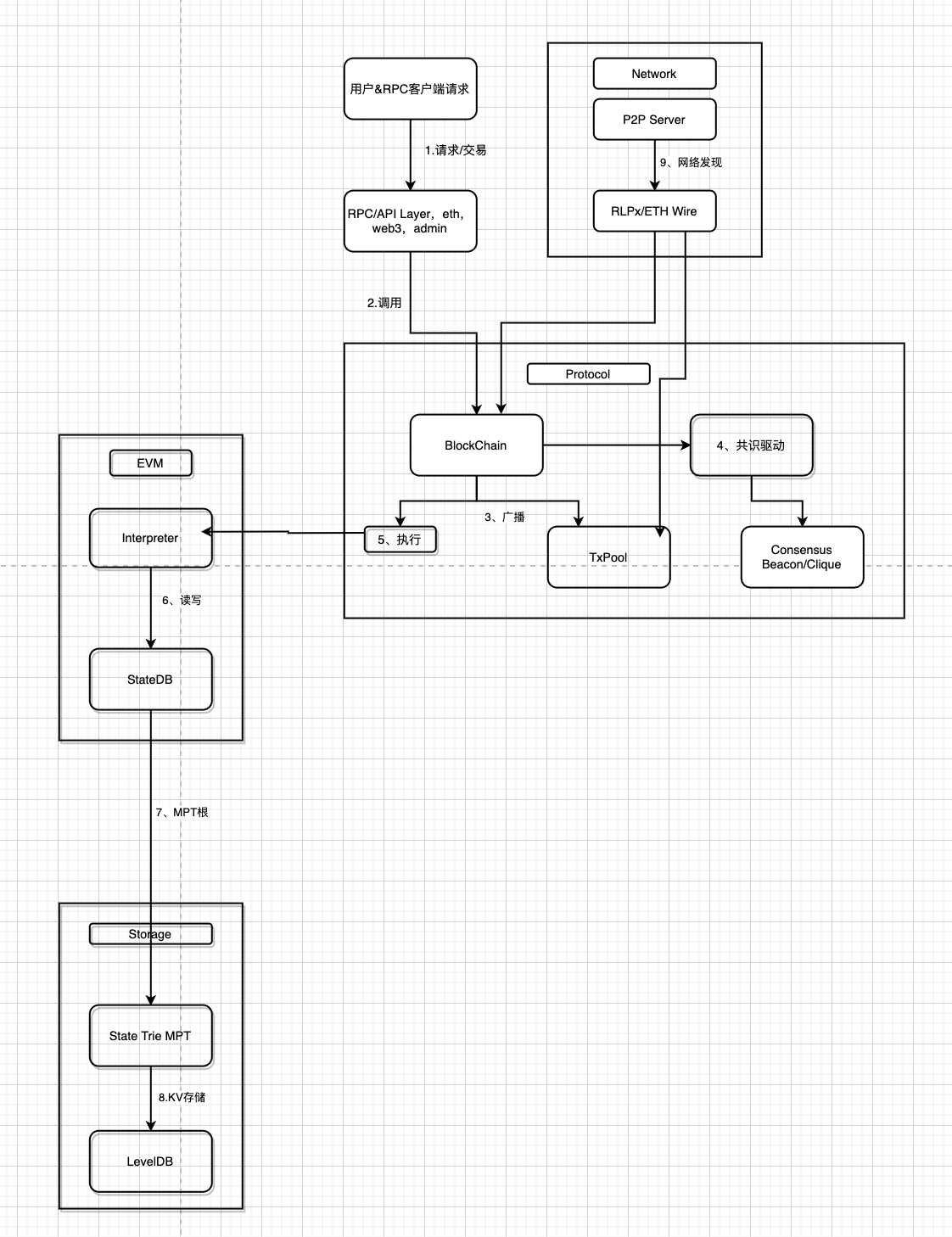
6、RPC/外部接口

对外提供账户余额查询、交易广播、合约调用等功能。

交流流程图



分层架构图如下：



1、用户 / RPC 客户端 把请求（查询余额、发交易）发给 RPC / API 层。

2、RPC 层 把请求转给 BlockChain，让它去验证、排队、打包。

3、BlockChain 把待打包交易丢给 TxPool，同时让 Consensus（Beacon 或 Clique） 决定下一个区块。

4、区块准备好后，BlockChain 调 Interpreter（EVM） 执行交易。

5、Interpreter 读写 StateDB，StateDB 再操作 State Trie（MPT） 来更新账户状态。

6、State Trie 把最终数据落到 LevelDB 做持久化。

7、所有节点通过 P2P Server + RLPx/ETH Wire 把区块、交易广播给全网，从而完成同步。

各层级关键模块：

1、 P2P 网络层

RLPx 负责加密握手、帧格式、多路复用

LES（Light Ethereum Subprotocol） 轻节点协议，让手机/浏览器只下载区块头 + 必要状态，不保存完整链数据

2、 区块链协议层

core/types 定义 区块、交易、收据、Header 等一切链上数据结构

core/blockchain 按序插入区块、维护侧链、触发分叉回滚

core/txpool 暂存未上链的交易，提供优先级队列

consensus/beacon PoS 共识 引擎，连接 Beacon 链，决定下一个区块

3 、EVM 执行层

core/vm/interpreter 逐条执行 EVM 字节码

core/state/state\_object 账户对象（余额、nonce、存储）

core/state/statedb StateDB 缓存层，读写后统一刷到 MPT

4、状态存储层

trie Merkle Patricia Trie 实现，负责 账户树、存储树、交易树、收据树 的增删改查

trie/database 把 trie 节点缓存 + 预写日志 (Preimage) 交给 LevelDB

ethdb/leveldb LevelDB 封装，真正落盘

5、 RPC & 轻节点层

les 轻节点客户端，仅拉区块头 + 状态证明，依赖 LES 协议

ethclient 对外暴露 eth\_getBalance, eth\_call 等 JSON-RPC

总结就是：RLPx 把节点连起来 → core/types 定义数据结构 → txpool/区块链层打包 → interpreter 执行 → StateDB + trie 更新 → LevelDB 落盘；LES 让手机也能用。