L05: Ethereum & Smart Contracts

Extended Slides – BSc Blockchain Course

Digital Finance

Outline

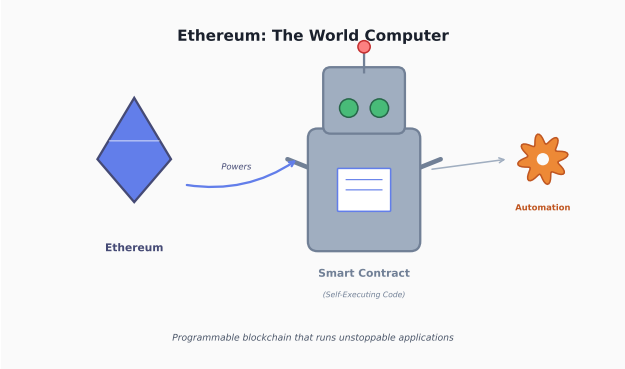
Learning Objectives

By the end of this lesson, you will be able to:

1. Compare Ethereum's account model with Bitcoin's UTXO model
2. Explain the Ethereum Virtual Machine (EVM) architecture
3. Understand gas mechanics and EIP-1559 fee market
4. Describe smart contract lifecycle and storage layout
5. Analyze Ethereum's state management via tries

Prerequisites: L03 Bitcoin Deep Dive, L04 Consensus Mechanisms.

Programmable Money



Purpose: Ethereum extended blockchain from simple payments to programmable applications. Smart contracts enable DeFi, NFTs, and decentralized organizations.

The platform that launched an entire ecosystem of decentralized applications.

Ethereum's Vision

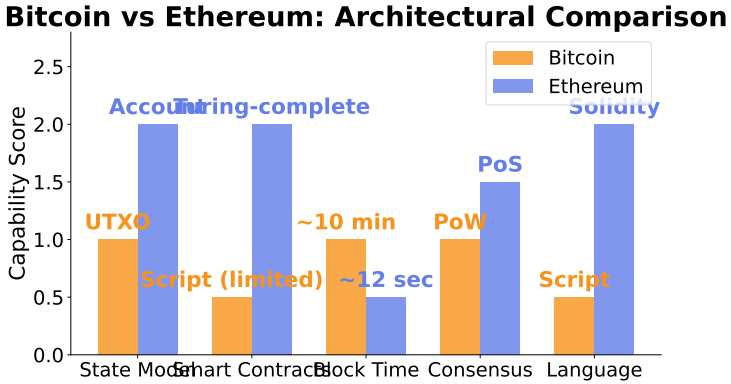
Beyond Digital Cash:

- Bitcoin: programmable money (limited Script)
- Ethereum: programmable state machine
- Smart contracts: self-executing agreements
- "World Computer" concept

Key Innovation:

- Turing-complete execution environment
- Persistent state storage
- Composable applications (DeFi, NFTs)

Vitalik Buterin proposed Ethereum in 2013, launched 2015.



Different design goals lead to different architectural choices.

UTXO vs Account Model

Bitcoin (UTXO):

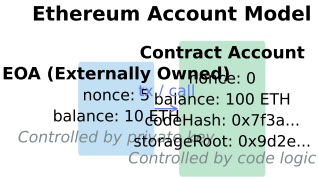
- Unspent Transaction Outputs
- Stateless verification
- Better privacy (new addresses)

Ethereum (Account):

- Balance stored in account state
- Simpler mental model
- Enables complex contract state

Account model trades some privacy for programmability.

Account Types



Account State Components:

- nonce: transaction count (EOA) / contracts created (CA)
- balance: Wei held by account
- codeHash: hash of contract bytecode (only CA)

Every account has nonce and balance; contracts also have code and storage.

Account State Fields

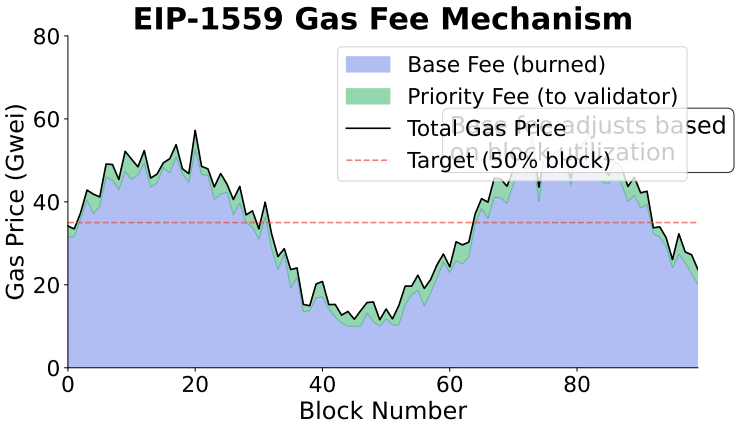
All Accounts Have:

- **nonce**: transaction count (EOA) or contracts created (CA)
- **balance**: Wei held (1 ETH = 10^{18} Wei)

Contract Accounts Also Have:

- **codeHash**: keccak256 hash of EVM bytecode
- **storageRoot**: root of storage trie

EOA codeHash is hash of empty string; storageRoot is empty trie root.



Base fee burned; priority fee to validators.

EIP-1559 Fee Market

Before EIP-1559 (Legacy):

- First-price auction: highest bidder wins
- Unpredictable fees, overpayment common

After EIP-1559:

- Base fee: algorithmically determined
- Adjusts up/down based on block utilization
- Target: 50% block capacity (15M gas)
- Priority fee (tip): incentive for inclusion

Base fee increases 12.5% per block when full.

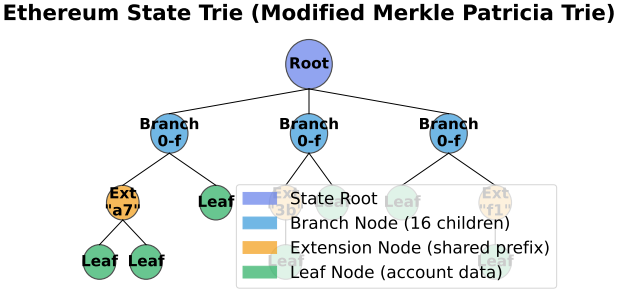
Gas Costs by Operation

Operation Costs (gas units):

- Gtransaction: 21,000 (base tx cost)
- Gcreate: 32,000 (contract creation)
- SSTORE (new): 20,000
- SSTORE (update): 5,000
- SLOAD: 2,100
- Memory expansion: quadratic

Storage is expensive by design to prevent state bloat. Gas costs shown are for cold access; warm access is cheaper.

State Trie Structure



Modified Merkle Patricia Trie optimizes for sparse key spaces.

Why Merkle Patricia Trie?

Requirements:

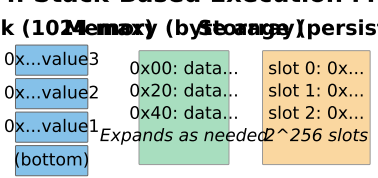
- Efficient key-value lookups
- Cryptographic commitments (state root)
- Support for light client proofs

Trie Types in Ethereum:

- State trie: all account states
- Storage trie: per-contract storage
- Transaction trie: per-block transactions
- Receipt trie: per-block receipts

State root in block header commits to entire world state.

EVM: Stack-Based Execution Model



Common EVM Opcodes:

PUSH	ADD/MUL	MSTORE	SSTORE	CALL
Stack ops	Arithmetic	Memory	Storage (20k gas)	External call

Gas Cost: Stack ops (3) < Memory (3+) < Storage (20000 write)

Stack-based VM with 1024-item stack limit.

EVM Properties

Deterministic Execution:

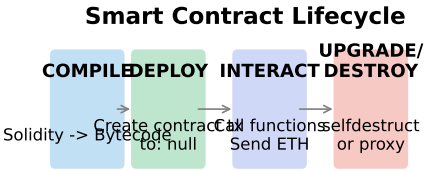
- Same input always produces same output
- No external randomness (use VRF/oracles)
- All nodes compute same result

Isolation:

- Contracts cannot access filesystem
- No network calls (use oracles)
- Limited to blockchain state

Determinism enables trustless verification.

Contract Lifecycle



Key Concepts:

- Constructor runs ONCE at deployment (initialization)
- Contract address = keccak256(deployer, nonce)
- Code is immutable after deployment (unless proxy pattern)
- selfdestruct sends remaining ETH to specified address

Constructor runs once; code immutable thereafter.

Contract Deployment

Deployment Process:

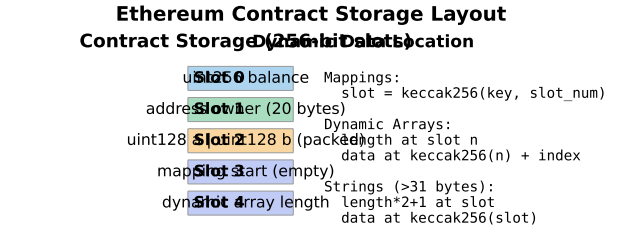
1. Compile Solidity to bytecode
2. Send transaction with to=null
3. EVM executes init code (constructor)
4. Returns runtime bytecode
5. Stored at new address

Address Calculation:

- CREATE: keccak256(deployer, nonce)
- CREATE2: keccak256(0xff, deployer, salt, codeHash)

CREATE2 enables deterministic addresses across chains.

Storage Layout



Storage: Most expensive EVM operation (SSTORE = 20,000 gas for new value)

Understanding layout crucial for upgradeable contracts.

Storage Slot Rules

Fixed-Size Variables:

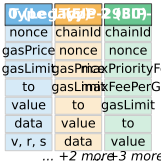
- Assigned slots in declaration order
- Packed if fit in 32 bytes together
- uint128 + uint128 = 1 slot

Dynamic Types:

- Mapping: slot = keccak256(key, slot_num)
- Dynamic array: length at slot, data at keccak256(slot)
- String/bytes: depends on length

Slot collisions can corrupt data in proxies.

Transaction Types



Type 2 (EIP-1559) now standard for most transactions.

Transaction Fields Explained

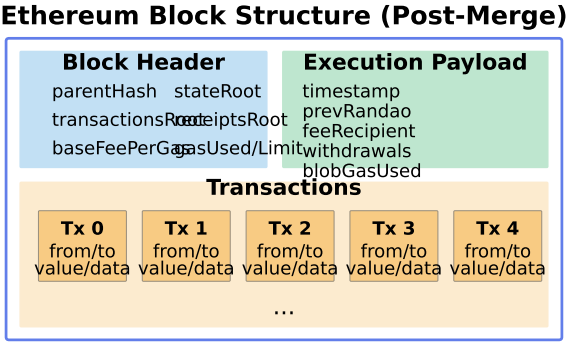
Common Fields:

- nonce: prevents replay attacks
- to: recipient (null for deployment)
- value: Wei to transfer
- data: function call or init code

EIP-1559 Specific:

- maxFeePerGas: absolute max willing to pay
- maxPriorityFeePerGas: tip to validator

Actual fee = min(baseFee + priorityFee, maxFee).



Post-Merge blocks have execution payload from PoS.

Block Header Fields

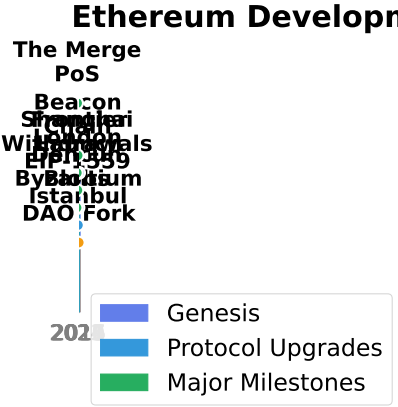
Key Fields:

- parentHash: link to previous block
- stateRoot: world state commitment
- transactionsRoot: tx trie root
- receiptsRoot: receipt trie root
- baseFeePerGas: current base fee

Post-Merge Additions:

- prevRandao: randomness from beacon
- withdrawals: validator withdrawals

Block proposer selected by beacon chain.



Future

Surge (sharding)

Verge (verkle)

Purge (history)

Splurge (misc)

Scaling now via L2; base layer for security and data availability.

The Merge (2022)

What Changed:

- Consensus: PoW to PoS
- Energy: 99.95% reduction
- Issuance: reduced significantly
- Block time: fixed 12 seconds

What Stayed Same:

- EVM execution model
- Transaction format
- Smart contract compatibility

Seamless transition – no contract changes required.

Future: Surge, Verge, Purge, Splurge

The Surge: Scalability via sharding

- Data availability sampling
- 100k TPS target with L2s

The Verge: Verkle trees

- Smaller proofs, stateless clients

The Purge: History expiry

- Remove old state obligations

The Splurge: Misc improvements

Multi-year roadmap; L2s handle scaling now.

Key Takeaways

Remember These Points

1. Account model: simpler than UTXO, enables complex state
2. EVM: deterministic, stack-based, gas-metered
3. EIP-1559: predictable fees, ETH burning
4. State: Merkle Patricia Tries for efficient proofs
5. Storage: 256-bit slots, expensive writes
6. Roadmap: L2 scaling, future verkle/sharding

Next Lesson: Solidity Programming – writing smart contracts.