Звіт з виконання лабораторної роботи 2
Тип 2
"Оцінка та обґрунтування необхідних ресурсів (газу і ефіру), потрібних для функціонування смарт-контракту "

ФІ-42мн Бондар Петро
ФІ-42мн Кістаєв Матвій

19 червня 2025 р.

Мета: Отримання теоретичного розуміння та навичок роботи із смарт-контрактами.

Задача: Дослідити принципи роботи смарт-контрактів, а також їх вартості (газ, ефір) в мережі Ethereum.

Що таке смарт-контракт?

Смарт-контракт -- це програмована логіка, яка виконується на блокчейні при настанні заданих умов або за викликом користувача. Смарт-контракти забезпечують прозоре, незмінне та децентралізоване виконання логіки, на основі консенсусу, без потреби у посередниках-гарантах.

Програмний код смарт-контракту зберігається за унікальною адресою в блокчейні та виконується віртуальною машиною EVM (Ethereum Virtual Machine) на кожному вузлі мережі. Усі дії контракту відбуваються публічно у вигляді транзакцій в блоці мережі, з перевіркою усіма вузлами мережі.

Оскільки смарт-контракти є Тюринг-повними, вони дозволяють створювати довільні децентралізовані додатки (dApps), токени, фінансові протоколи (DeFi) та інші форми цифрової взаємодії без довіри до центральної сторони.

Історія

Ідея смарт-контрактів була вперше запропонована Ніком Сабо в 97 році, як концепція переносу економічних процесів та державного регулювання на комп'ютерні мережі, незалежні від людей. Проте тоді ще не йшлось про будь-які консенсусні розподілені обчислення.

В 2014 році Віталік Бутерін в Ethereum White paper, на основі концепції блокчейна для обробки криптовалюти, запропонував узагальнену концепцію блокчейну, який також може розподілено виконувати Тюринг-повні обчислення, результат яких визначатиметься консенсусом.

Із 2015 року ця концепція смарт-контрактів була запроваджена в мережі Ethereum.

Структура контракту

Код смарт-контракту містить функції: конструктор, який викликається під час публікації контракту, публічні функції, які можуть будути викликані довільним користувачем, та внутрішні допоміжні функції, які викликаються всередині публічних функцій.

Таким чином контракт може виконувати довільні обчислення всередині себе, а також взаємодіяти із мережею -- наприклад, викликати функції в інших контрактах, або публікувати транзакції в чергу.

Смарт-контракти в Ethereum мають власне постійне сховище, яке дозволяє зберігати дані між викликами функцій. Це сховище організоване у вигляді словника ключ–значення, де ключі та значення — це 256-бітні слова. Зберігання відбувається на ланцюгу, і кожен вузол мережі підтримує актуальну копію цього сховища для кожного контракту. Тому використання пам'яті смарт-контракту є доволі вартісним.

Вся пам'ять контракту зберігається у вигляді піддерева глобального дерева Меркле-Патрації, яке зберігає весь стан ланцюга, і кожен вузол мережі має свою локальну копію цього дерева.

Також, кожен контракт має свій баланс. Таким чином контракт може зберігати/отримувати/передавати ефір.

Виконання контракту

Смарт-контракти в Ethereum виконуються за принципом детермінованої децентралізованої обробки, що забезпечує однаковий результат на кожному вузлі мережі.

Виконання контракту ініціюється через транзакції та обробляється Ethereum Virtual Machine (EVM), використовуючи байткод контракту.

1. Ініціація транзакції
    - Користувач або інший контракт надсилає транзакцію до адреси контракту.
    - Транзакція має спеціальний формат: адресу контракту, закодований виклик функції, аргументи, газліміт, вартість газу та цифровий підпис.
2. Додавання до блоку
    - Вузол-пропозер включає транзакцію до чергового блоку.
    - Усі зміни стану, спричинені транзакцією, впливають на глобальний стан.
3. Виконання віртуальною машиною
    - Кожен повноцінний вузол виконує байткод контракту локально через EVM.
    - Виклик функції змінює пам'ять контракту, баланси, логи або генерує інші події.
4. Валідація стану
    - Результат виконання змінює кореневий геш дерева пам'яті контракту `storageRoot`, а отже і глобального стану -- `stateRoot`.
    - Якщо обчислений хеш не збігається з вказаним у блоці — вузол відкидає блок як невалідний.

Приклад

```solidity
pragma solidity ^0.8.0;

contract OwnedStorage {
    address public owner;
    uint256 private value;

    constructor() {
        owner = msg.sender;
    }

    // Дозволяє встановити значення змінної лише власнику
    function set(uint256 _value) public {
        require(msg.sender == owner, "Only owner can set value");
        value = _value;
    }

    // Дозволяє переглянути значення змінної будь-кому
    function get() public view returns (uint256) {
        return value;
    }
}
```

Listing 1: Простий контракт на Solidity, який зберігає одну змінну в пам'яті та має метод для встановлення значення цієї змінної, який працює тільки для власника, який опублікував цей контракт.

Вартість роботи смарт-контрактів

Будь-які операції EVM мають свою вартість у вигляді газу, таким чином, валідуючий вузол, який "проводить" довільний блок, вимірює кількість використаних для цього ресурсів.

Коли валідатор проводить блок в ланцюгу, він бере комісію із транзакцій в блоці відповідно до кількості використаного газу. А оскільки логіка смарт-контракту виконується в результаті виклику функції через транзакцію спеціального формату, то і розмір комісії, яка стягується, залежить від складності внутрішньої логіки контракту.

Category	Operation	Gas Cost (Gwei)
Arithmetic	ADD, SUB	3
	MUL, DIV, MOD	5
	EXP (exponentiation)	10 + cost per byte
Storage Access	SLOAD (read storage)	2,100
	SSTORE (write new value)	20,000
	SSTORE (change non-zero to zero)	5,000 (with refund)
	SSTORE (overwrite same value)	100
Memory Access	MLOAD, MSTORE	3
Control Flow	JUMP, JUMPI	8
	PC, GAS	2
Call / External	CALL	700 + dynamic cost
	DELEGATECALL, STATICCALL	700 + dynamic cost
	CREATE	32,000 + code cost
	CALLDATACOPY, CODECOPY	3 + 3 per word
Logs / Events	LOG	375 + 8 per byte
Miscellaneous	BALANCE, EXTCODESIZE	100
	BLOCKHASH	20
	SELFDESTRUCT	5,000 (subject to refund)
	RETURN	0 (plus memory cost)

Табл. 1: Приблизні ціни на газ для базових операцій EVM

Як можна побачити із таблиці, найдорожчими операціями є операції із внутрішньою пам'ятю контракту, а також операції зовнішньої взаємодії із блокчейном.

Приклад 1: ERC-20

ERC-20 — це стандарт смарт-контрактів для створення інших взаємозамінних токенів на блокчейні Ethereum, який визначає набір функцій які має містити контракт і правил за якими він має оперувати. Стандарт забезпечує сумісність між різними смарт-контрактами, гаманцями, біржами та децентралізованими застосунками.

Основні функції стандарту:

- перевірку балансу,
- переказ токенів,
- делегування прав на переказ і перевірку дозволу

Відомими прикладами ERC-20 токенів є USDT, DAI, LINK та UNI.

Розглянемо, наприклад, операцію `transfer()` для ERC-20 токена:

Компонент	**Опис**	**Витрати газу**
Базова вартість транзакції	Постійна вартість будь-якої транзакції в Ethereum	21 000
`SLOAD` балансу відправника	Зчитування балансу зі сховища	2 100
`SLOAD` балансу одержувача	Зчитування балансу одержувача	2 100
Арифметика	Додавання/віднімання, перевірки	приблизно 500
`SSTORE` балансу відправника	Запис оновленого балансу	5 000
`SSTORE` балансу одержувача	Запис оновленого балансу	5 000
`Transfer` подія (логування)	Створення події з 2 темами та даними	приблизно 4 000
Інші витрати	Стек, виклики функцій, перевірки	2 000
Загалом (приблизно)		**35 000–40 000**

Табл. 2: Оцінка витрат газу на виконання `transfer()` ERC-20 токена

Приклад 2: UniSwap

Uniswap — це децентралізована біржа (DEX) на блокчейні Ethereum, яка дозволяє користувачам обмінювати токени *безпосередньо зі своїх гаманців*, без посередників. Це один із найвідоміших протоколів у сфері DeFi (децентралізованих фінансів).

Це повністю смарт-контрактна модель -- вся логіка реалізована у вигляді відкритих смарт-контрактів.

Крок	Компонент	Приблизна вартість газу
1	Базова вартість транзакції	21 000
2	Виклик `approve()` для токена (якщо ще не було)	45 000 – 50 000
3	Виклик `swapExactTokensForTokens()` (логіка роутера)	100 000 – 160 000
4	Внутрішній виклик `UniswapV2Pair.swap()`	80 000 – 110 000
5	ERC-20 `transferFrom` (токен від користувача до пулу)	35 000 – 40 000
6	ERC-20 `transfer` (токен від пулу до користувача)	35 000 – 40 000
7	Логування подій (Transfer, Swap, Sync і т.п.)	10 000 – 15 000
	Загальна оцінка без урахування `approve()`	**140 000 – 200 000**

Табл. 3: Приблизні витрати газу для обміну токенів на Uniswap V2

Висновки

Смарт-контракти — це програми, які зберігаються та виконуються у блокчейні, зокрема в мережі Ethereum. Вони автоматизують виконання угод між сторонами без потреби у посередниках. Смарт-контракти мають власну адресу та стан (дані), а також зберігають свій програмний код. При виклику контракту кожен вузол мережі виконує його код локально, щоб обчислити новий стан і перевірити його відповідність. Таким чином, виконання контракту є детерміністичним і підлягає досягненню консенсусу всіма учасниками мережі.

Смарт-контракти дозволяють будувати довільні розподілені обчислення на блокчейні, найбільш цікавими є розподілені програми та розподілені фінансові протоколи.

Виконання смарт-контрактів потребує споживання ресурсу, який називається газ. Кожна операція у віртуальній машині Ethereum (EVM) має певну вартість у газі: прості дії, як додавання чи перевірка умов, коштують менше, тоді як запис до сховища або створення контракту — значно дорожче. Обробка контрактів коштує доволі багато газу, особливо використання внутрішньої пам'яті контракту. Тому до дизайну і програмування контрактів треба підходити дуже ретельно.