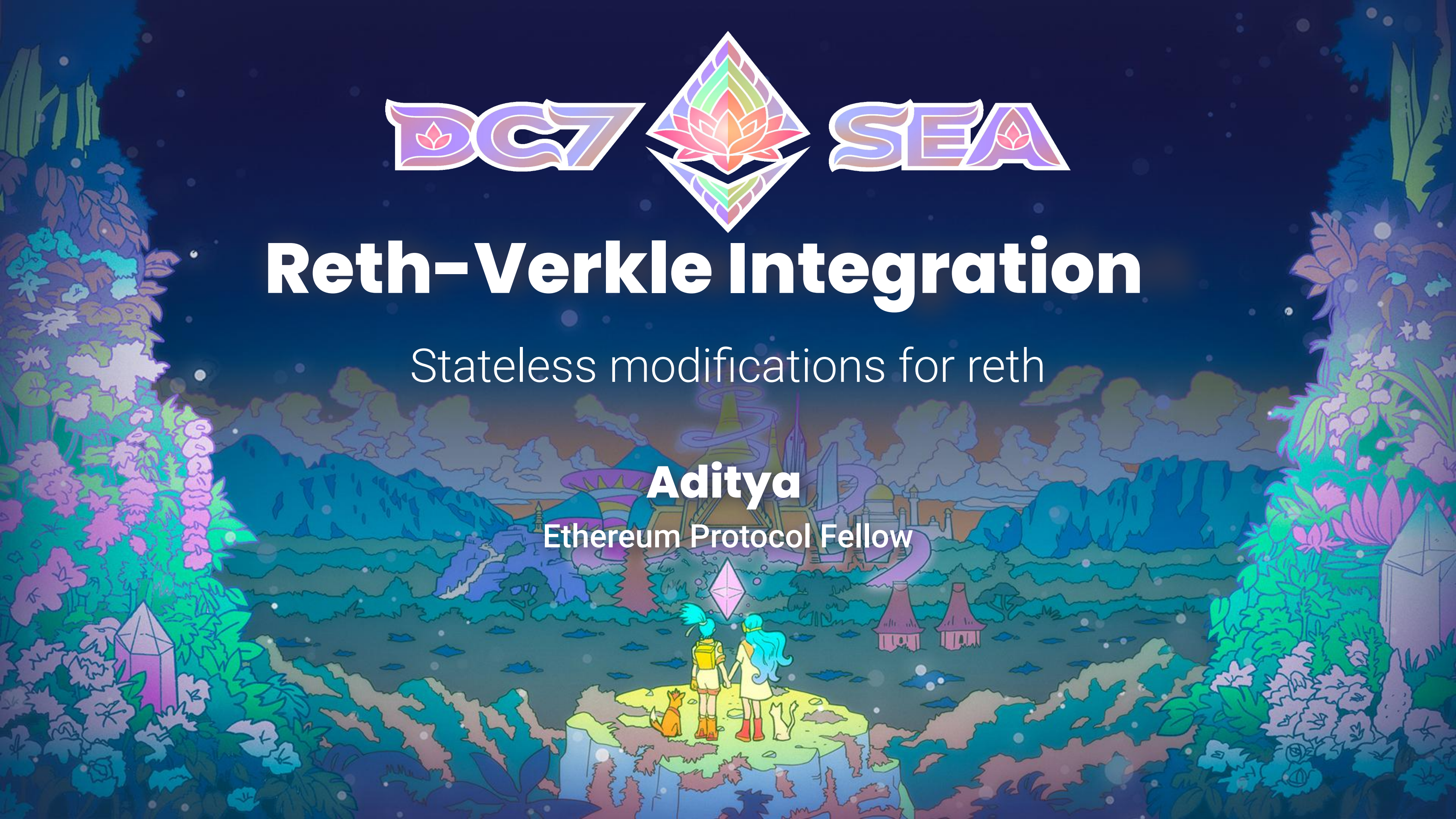
Reth-Verkle Integration

Stateless modifications for reth

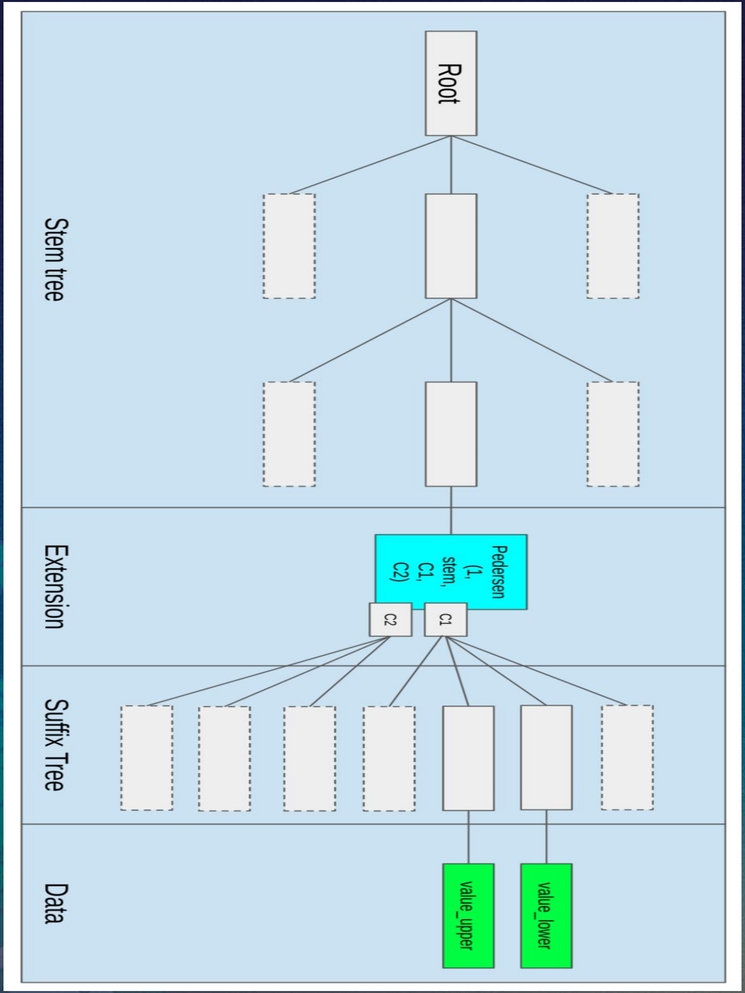
Aditya

Ethereum Protocol Fellow



Section 1

changes in state commitment structure



EIP-6800: Ethereum state using a unified verkle trie

Instead of a two-layer structure as in the Patricia tree, in the Verkle tree we will embed all information into a single key: value tree

Parameter	Value
BASIC_DATA_LEAF_KEY	0
CODE_HASH_LEAF_KEY	1
HEADER_STORAGE_OFFSET	64
CODE_OFFSET	128
VERKLE_NODE_WIDTH	256
BASIC_DATA_LEAF_KEY	256**31

changes in state commitment structure

- Previously different tries were needed to be maintained: Account Trie and Storage Trie.
- Now all data required for state commitment were present in unified verkle trie.
- Earlier: ***StateProvider: StateRootProvider*** + ***StorageRootProvider***
- Now: ***VerkleRootProvider***
- State root computations in Post-transaction, Block Finalization, State root verification, chain reorgs.
- Caching, Incremental updates, parallelization.
- Database modifications required: **single table structure for current state**

Section 2

EIP-4762

EIP-4762: Statelessness gas cost changes

- Data access is divided into two types: AccessEvents and WriteEvents
- Different access patterns for Account Data, Storage slots, contract code and transactions.
- Describes changes in gas costs while dealing with various opcodes, precompiles and system contracts.
- Opcodes affected: CALL, CALLCODE, DELEGATECALL, STATICCALL, CREATE, CREATE2, SLOAD, SSTORE, EXTCODESIZE, EXTCODECOPY, EXTCODEHASH, CODECOPY, CODESIZE, BALANCE, SELFDESTRUCT, JUMP, JUMP1, PUSH1 through PUSH3.
- WITNESS_BRANCH_COST (1900) WITNESS_CHUNK_COST (200)
- SUBTREE_EDIT_COST (3000) CHUNK_EDIT_COST (500) CHUNK_FILL_COST (6200)

Revm implementation

- Different functions with different gas calculations logic for different opcodes.
- Each function internal calls ***access_key*** method responsible for gas calculations with altered parameters.
- All these functions are then called internally while executing particular opcodes.

access_account_data
access_code_hash
access_for_balance_op_code
access_for_storage
access_for_block_hash_op_code
access_for_code_program_counter
access_and_charge_for_code_slice
access_code_chunk
access_for_absent_account
access_for_self_destruct
access_for_contract_creation_check
access_for_contract_creation_init
access_for_contract_created
access_for_value_transfer
access_for_gas_beneficiary
access_account_for_withdrawal
access_for_blockhash_insertion_wit
ness
access_for_transaction
access_basic_data
access_code_hash_internal
access_complete_account
access_account_subtree

Section 3

Test vectors

Verkle execution spec tests

4762

- test_balance
- test_calls
- test_codecopy_ext_precompile
- test_codecopy_generic
- test_codecopy_generic_initcode
- test_coinbase_fees
- test_contract_execution
- test_creates
- test_extcodehash
- test_extcodesize
- test_selfdestruct
- test_sload
- test_sstore
- test_transfer
- test_withdrawls

6800

- test_contract_codechunking
- test_contract_creation
- test_storage_slot_write
- test_transfer

7709

- test_blockhash_instruction

Devnet 7 ?

Section 4

The plan ahead

Future milestones

Completed

- Basic level of state commitment modifications.
- 4762 gas cost changes in revm
- 6800 readiness of the trie.

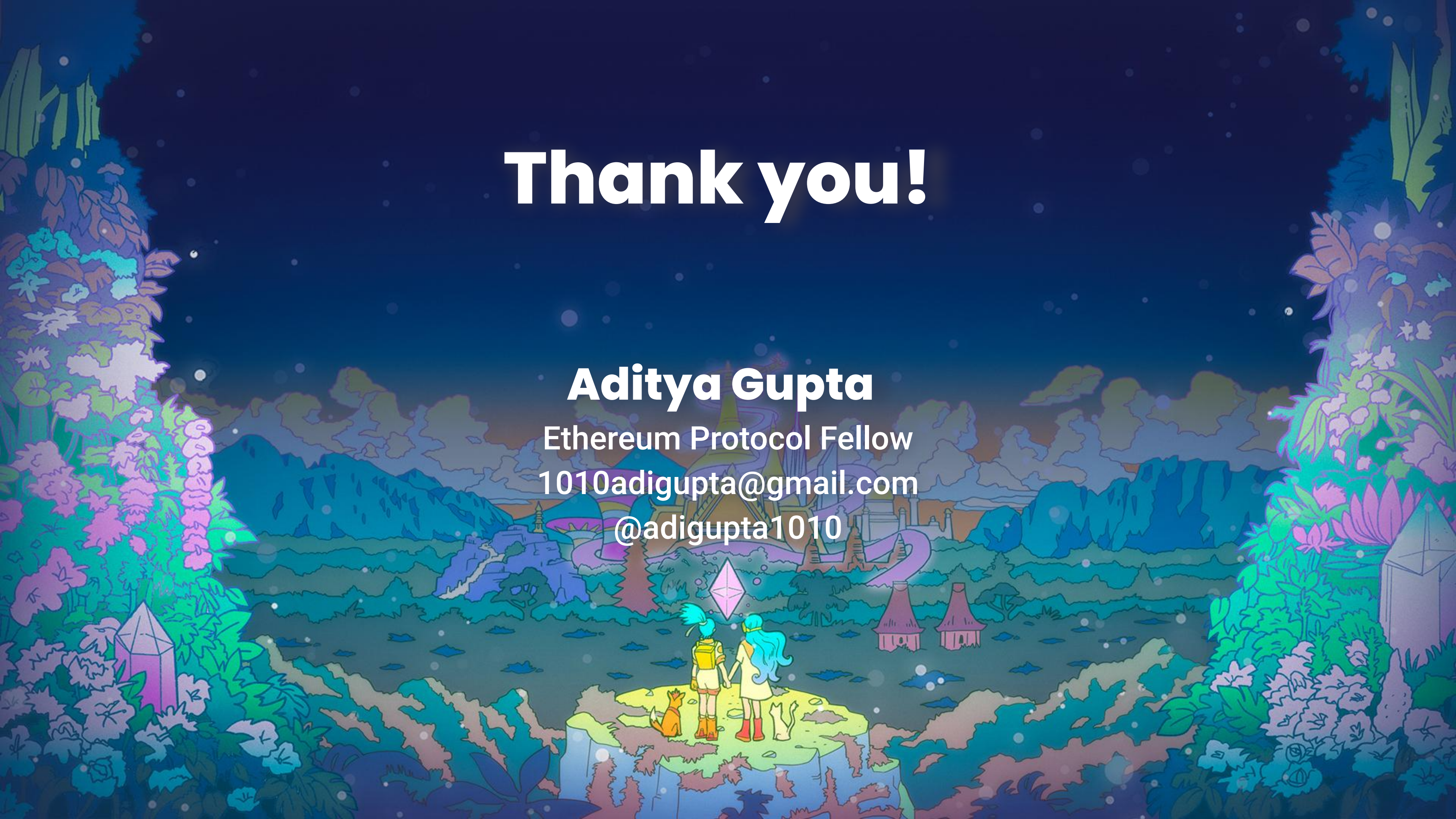
Next phase

- Modify other parts such networking stack, ssz, parallelizing root computation etc.
- **Stateless Execution using witness**

More into future

- Working on sync design and transition strategy with the stateless team.
- Exploring other trie design eg. binary hash trie for statelessness
- SNARKing Verkle, Circle STARKs, and STARKed binary hash trees

EPF !!!



Thank you!

Aditya Gupta
Ethereum Protocol Fellow
1010adigupta@gmail.com
@adigupta1010