SoCoSi

Solidity Cost Simulator

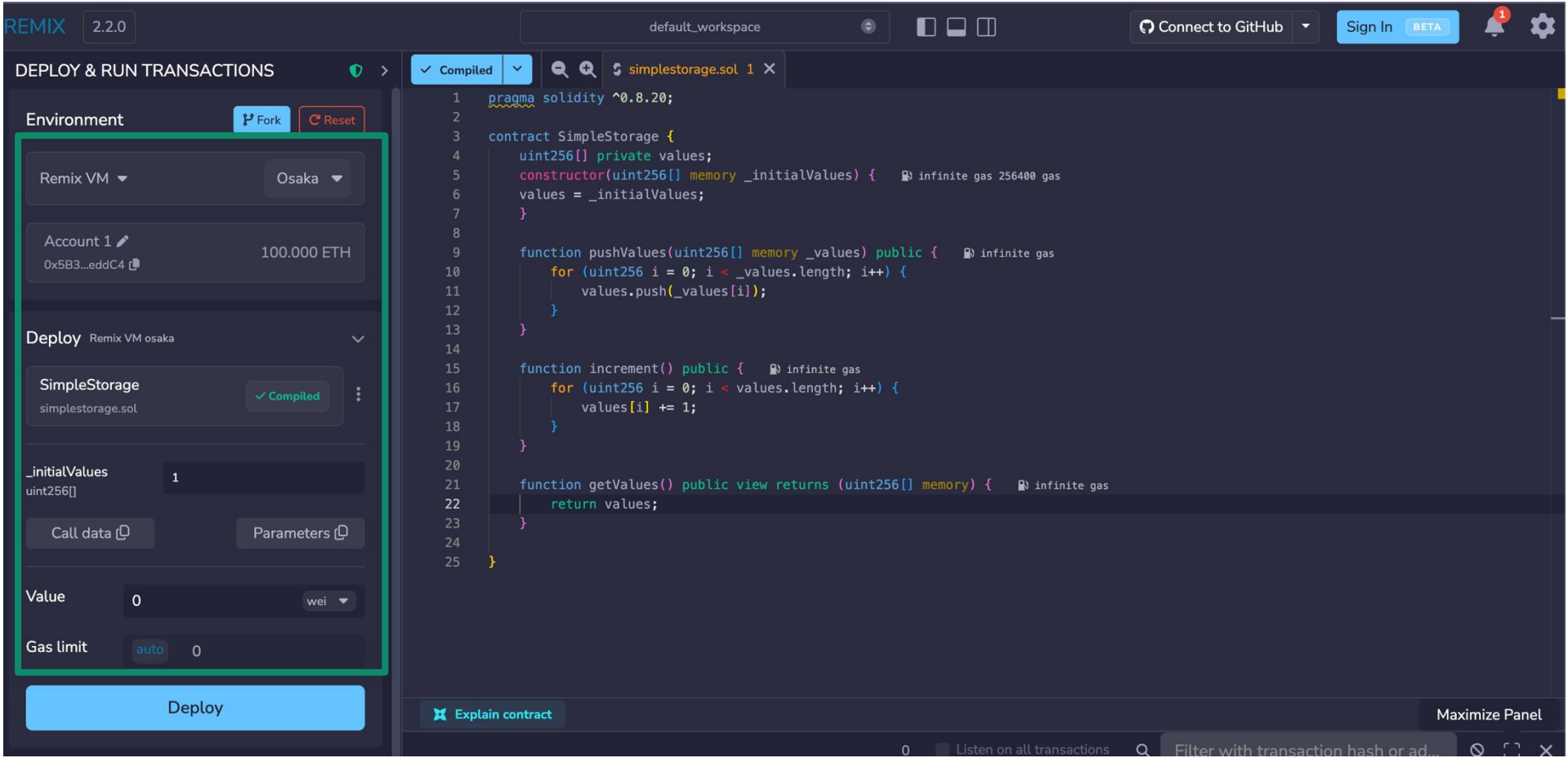
A Tool for Gas Cost Estimation
of Smart Contracts on Solidity-based Blockchains

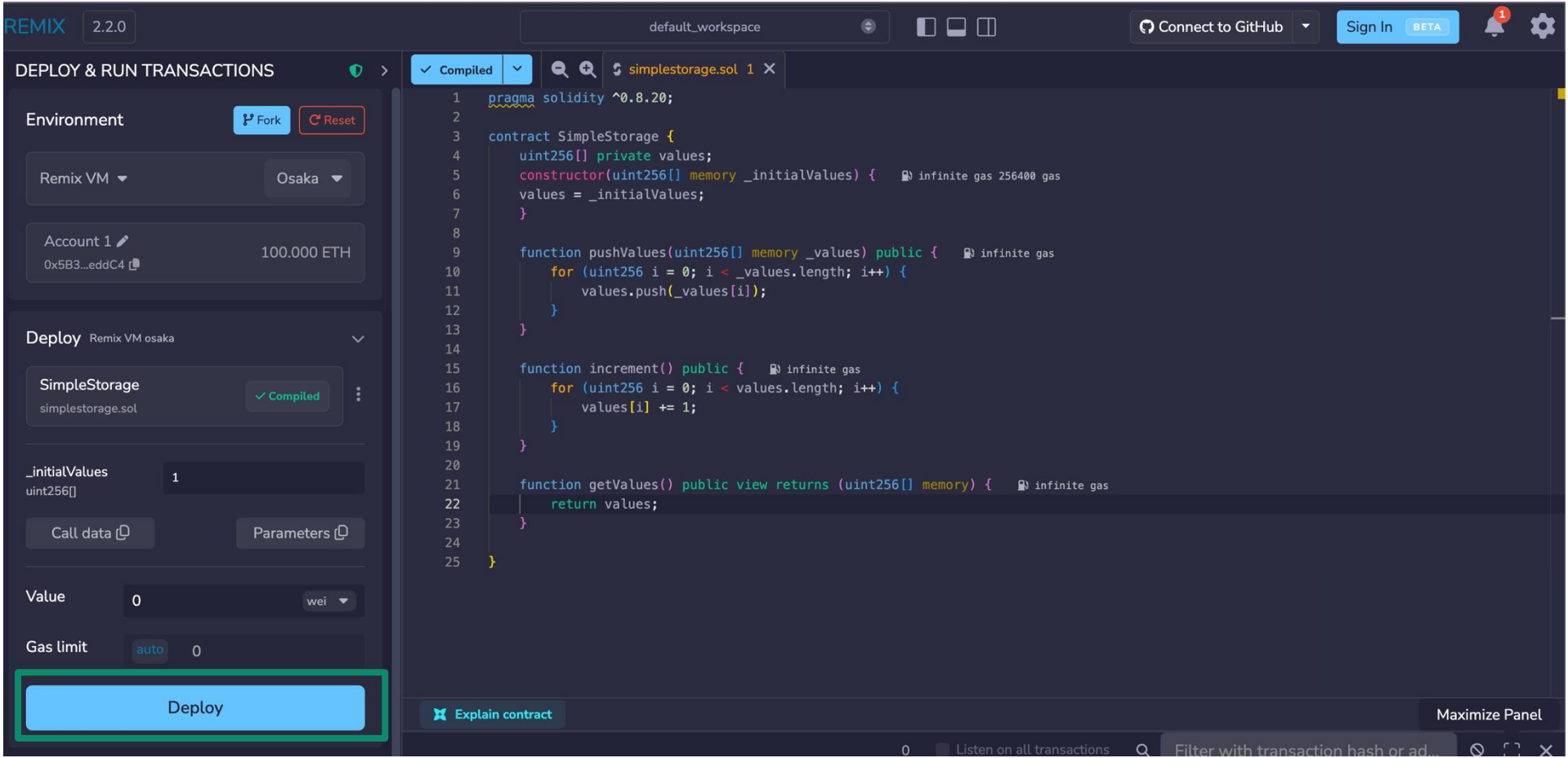
Stefano Bistarelli · **Ivan Mercanti** · Carlo Taticchi

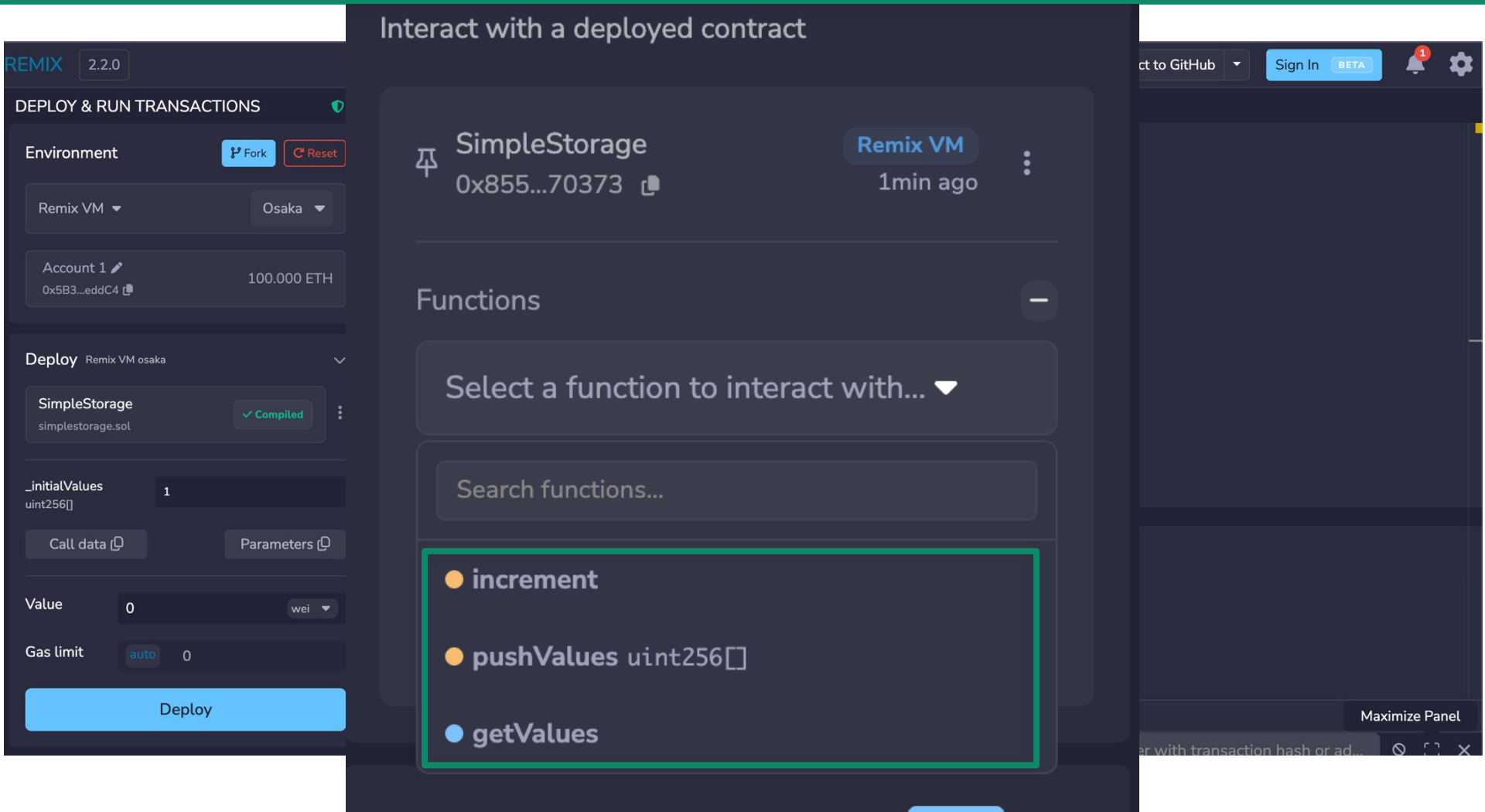
University of Perugia, Italy

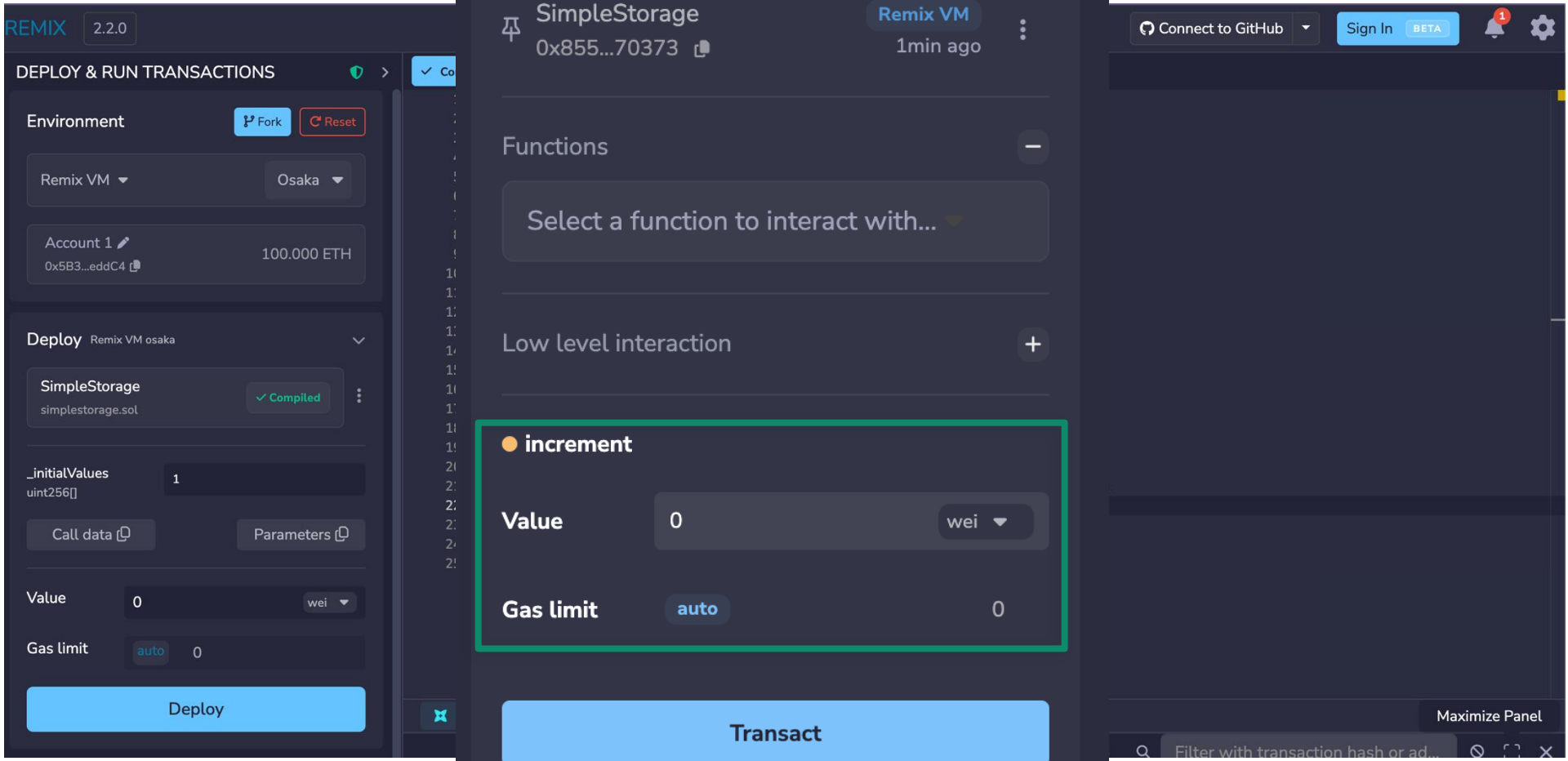
DLT Workshop · June 4–6, 2026 · Pula, Italy

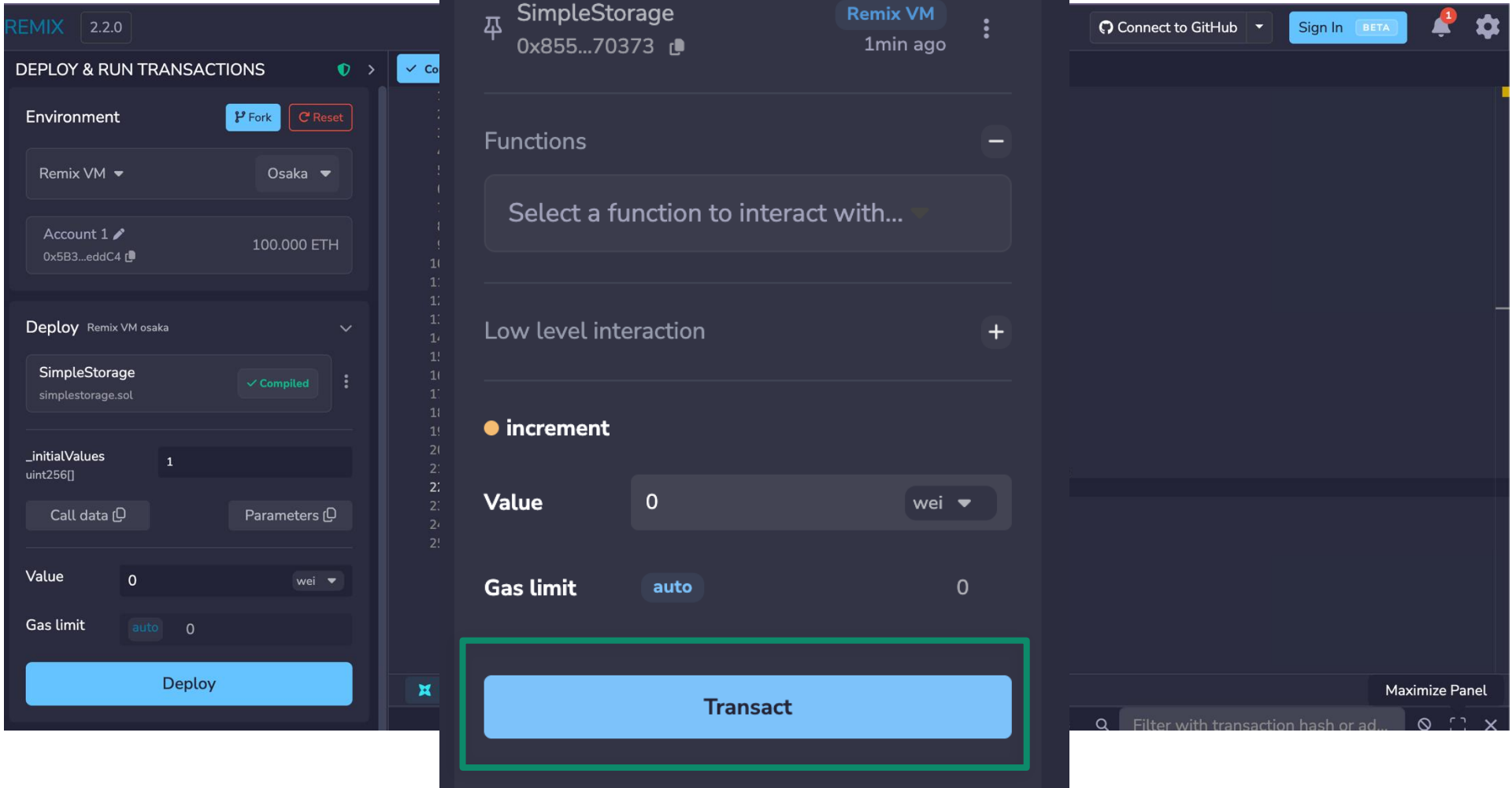
A.D. 1308
unipg
DIPARTIMENTO
DI MATEMATICA E INFORMATICA


REMIX
2.2.0
default_workspace
Connect to GitHub
Sign In
BETA
DEPLOY & RUN TRANSACTIONS
Environment
Fork
Reset
Remix VM
Osaka
Account 1
0x5B3...eddC4
100.000 ETH
Deploy
Remix VM osaka
SimpleStorage
simplestorage.sol
Compiled
_initialValues
uint256[]
1
Call data
Parameters
Value
0
wei
Gas limit
auto
0
Deploy
Compiled
simplestorage.sol 1
pragma solidity ^0.8.20;
contract SimpleStorage {
uint256[] private values;
constructor(uint256[] memory _initialValues) { infinite gas 256400 gas
values = _initialValues;
}
function pushValues(uint256[] memory _values) public { infinite gas
for (uint256 i = 0; i < _values.length; i++) {
values.push(_values[i]);
}
}
function increment() public { infinite gas
for (uint256 i = 0; i < values.length; i++) {
values[i] += 1;
}
}
function getValues() public view returns (uint256[] memory) { infinite gas
return values;
}
}
Explain contract
Maximize Panel
0
Listen on all transactions
Filter with transaction hash or ad...


REMIX
2.2.0
default_workspace
Connect to GitHub
Sign In
BETA
DEPLOY & RUN TRANSACTIONS
Environment
Fork
Reset
Remix VM
Osaka
Account 1
0x5B3...eddC4
100.000 ETH
Deploy
Remix VM osaka
SimpleStorage
simplestorage.sol
Compiled
_initialValues
uint256[]
1
Call data
Parameters
Value
0
wei
Gas limit
auto
0
Deploy
Compiled
simplestorage.sol 1
pragma solidity ^0.8.20;
contract SimpleStorage {
uint256[] private values;
constructor(uint256[] memory _initialValues) { infinite gas 256400 gas
values = _initialValues;
}
function pushValues(uint256[] memory _values) public { infinite gas
for (uint256 i = 0; i < _values.length; i++) {
values.push(_values[i]);
}
}
function increment() public { infinite gas
for (uint256 i = 0; i < values.length; i++) {
values[i] += 1;
}
}
function getValues() public view returns (uint256[] memory) { infinite gas
return values;
}
}
Explain contract
Maximize Panel
0
Listen on all transactions
Filter with transaction hash or ad...


Interact with a deployed contract
REMIX 2.2.0
DEPLOY & RUN TRANSACTIONS
Environment
Fork
Reset
Remix VM
Osaka
Account 1
0x5B3...eddC4
100.000 ETH
Deploy Remix VM osaka
SimpleStorage
simplestorage.sol
Compiled
_initialValues
uint256[]
1
Call data
Parameters
Value
0
wei
Gas limit
auto
0
Deploy
SimpleStorage
0x855...70373
Remix VM
1min ago
Functions
Select a function to interact with...
Search functions...
increment
pushValues uint256[]
getValues
Sign In
BETA
Maximize Panel


REMIX 2.2.0
DEPLOY & RUN TRANSACTIONS
Environment
Fork
Reset
Remix VM
Osaka
Account 1
0x5B3...eddC4
100.000 ETH
Deploy Remix VM osaka
SimpleStorage
simplestorage.sol
Compiled
_initialValues
uint256[]
1
Call data
Parameters
Value
0
wei
Gas limit
auto
0
Deploy
SimpleStorage
0x855...70373
Remix VM
1min ago
Functions
Select a function to interact with...
Low level interaction
increment
Value
0
wei
Gas limit
auto
0
Transact
Connect to GitHub
Sign In
BETA
Maximize Panel
Filter with transaction hash or ad...


REMIX
2.2.0
DEPLOY & RUN TRANSACTIONS
Environment
Fork
Reset
Remix VM
Osaka
Account 1
0x5B3...eddC4
100.000 ETH
Deploy
Remix VM osaka
SimpleStorage
simplestorage.sol
Compiled
_initialValues
uint256[]
1
Call data
Parameters
Value
0
wei
Gas limit
auto
0
Deploy
SimpleStorage
0x855...70373
Remix VM
1min ago
Functions
Select a function to interact with...
Low level interaction
increment
Value
0
wei
Gas limit
auto
0
Transact
Connect to GitHub
Sign In
BETA
Maximize Panel
Filter with transaction hash or ad...

[vm] **from:** 0x5B3...eddC4 **to:** SimpleStorage.increment() 0x855...70373 **value:** 0 wei **data:** 0xd09...de08a **logs:** 0 **hash:** 0x8e2...16441

Debug

status	1 Transaction mined and execution completed
transaction hash	0x8e28953d5010c7450a502e44d07a59ec1fc37ea9bcf4ce0d1af719a630016441
block hash	0xdb4c867e498d23466660817f8754544108e63bb4f0111cb068231bfd5c11ae05
block number	313
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	SimpleStorage.increment() 0x855659FE49EF3036Af43bb348C5D7B2566270373
transaction cost	28925 gas
execution cost	7861 gas
output	0x
decoded input	{}
decoded output	{}
logs	[]
raw logs	[]

0 Listen on all transactions Filter with transaction hash or ad...

The Problem

Gas = Money

Every EVM operation costs gas. Deployment and execution costs directly affect economic sustainability of dApps.

Limited Tooling

Existing tools rely on manual testing, ad-hoc scripts, or give only partial and approximate estimates.

No Lifecycle View

No support for ordered function sequences or full deployment + execution analysis in a single workflow.

Related Work

Tool	Deployment	Exec. Sequences	Full Lifecycle	Notes
GASOL	✓	✗	✗	Static analysis, single function only
sOptimize	✓	✗	✗	Bytecode optimization, no estimation
Remix IDE	✓	✗	✗	Approximate estimates only
Mythril / ZEUS	✗	✗	✗	Security analysis, not gas-focused
SoCoSi	✓	✓	✓	**Full lifecycle, scenario-based, reproducible**

Ethereum Gas Model

```
Transaction Cost (Gwei)  =  Gas Used  ×  Gas Price (Gwei)
```

Gas consumption depends on:

Contract Logic

Complexity of functions, branching, loops — all affect the EVM opcode count and cost.

Input Parameters

Different inputs trigger different execution paths, each with varying gas consumption.

Execution Context

State size, prior operations, and storage changes directly impact each call.

SoCoSi — System Model

Phase 1

Deployment

Constructor execution
Initial state s_0 established
Deployment gas measured

Phase 2

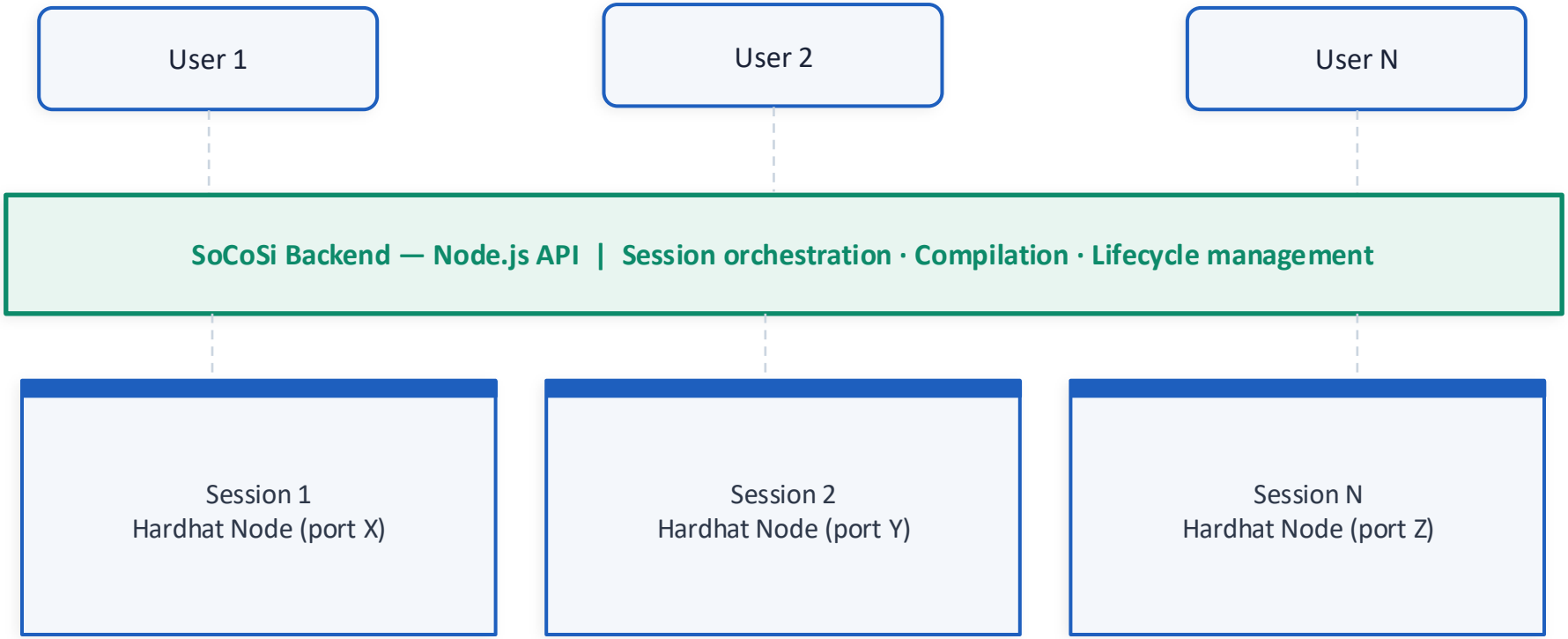
Execution

Ordered sequence $S = \{f_1, f_2, \ldots f_n\}$
State evolves with each call
Per-function gas recorded

Analysis Outputs

- Deployment Cost
- Per-function Gas
- Cumulative Cost
- State Evolution
- Path Comparison

System Architecture



Each session runs on a fully isolated Hardhat node — no state interference across users

Execution Pipeline

1	**Session Init**	Isolated Hardhat node spawned for each user request
2	**Contract Upload**	Solidity source code submitted and stored in session workspace
3	**Compilation**	Automatic compilation — ABI and bytecode artifacts generated
4	**Deployment**	Contract deployed with user-defined constructor parameters; deployment gas measured
5	**Sequential Execution**	Ordered function calls applied, state evolution tracked, per-function gas recorded
6	**Analysis Output**	Structured gas consumption report: per-function, cumulative, high-impact ops

Use Case: SimpleStorage Contract

SimpleStorage.sol

```
contract SimpleStorage {
  uint256[] private values;

  constructor(uint256[] memory _v) {
    values = _v;
  }

  function pushValues(uint256[] memory _v) {
    for (...) values.push(_v[i]);
  }

  function increment() {
    for (...) values[i] += 1;
  }

  function getValues() view returns (uint256[] memory) {
    return values
  }

}
```

Operation
Deployment (constructor([1]))
pushValues([10])
increment()
pushValues([20,30])
increment()
getValues()

Gas measured directly on a real EVM — not estimated. State evolution effects (growing array → higher increment cost) captured automatically.

Use Case: SimpleStorage Contract

SimpleStorage.sol

```
contract SimpleStorage {
  uint256[] private values;

  constructor(uint256[] memory _v) {
    values = _v;
  }

  function pushValues(uint256[] memory _v) {
    for (...) values.push(_v[i]);
  }

  function increment() {
    for (...) values[i] += 1;
  }

  function getValues() view returns (uint256[] memory) {
    return values
  }

}
```

Operation	Gas Used
Deployment (constructor([1]))	**402,030**
pushValues([10])	50,052
increment()	34,705
pushValues([20,30])	**73,063**
increment() — 2nd call	46,043
getValues()	*0 (read call)*
TOTAL	**605,893**

Gas measured directly on a real EVM — not estimated. State evolution effects (growing array → higher increment cost) captured automatically.

Conclusions & Future Work

Contributions

- Simulation-based framework for structured gas estimation
- Session-based isolation — reproducible, multi-user
- Ordered execution sequences capturing state dependencies
- Unified deployment + runtime cost analysis

Future Work

- Graphical User Interface (GUI)
- Export to LaTeX tables / Excel
- Predictive gas price models
- Probabilistic execution path modeling

SoCoSi makes gas cost analysis accessible, structured, and reproducible — enabling better design decisions for decentralized applications.

SoCoSi

QUESTION?

A Tool for Gas Cost Estimation
of Smart Contracts on Solidity-based Blockchains

Stefano Bistarelli · **Ivan Mercanti** · Carlo Taticchi

University of Perugia, Italy

DLT Workshop · June 4–6, 2026 · Pula, Italy

A.D. 1308
unipg
DIPARTIMENTO
DI MATEMATICA E INFORMATICA