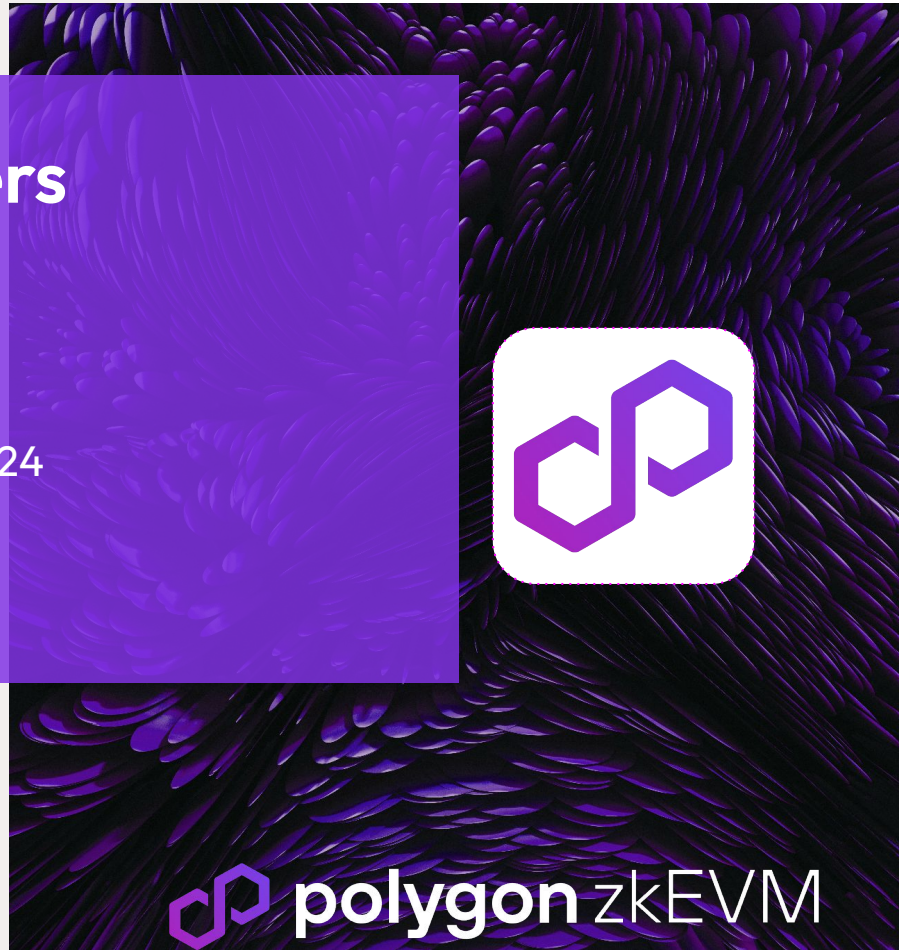
Top opcode offenders in the zkEVM

Bangkok, 13 November 2024
Devcon VII

Carlos Matallana Espinar
Protocol Team Lead



Presentation Outline

1. Intro
    - Zk-counters
    - Zk-counters types
    - Available zk-counters
2. Analysis offenders
    - Direct offenders
    - Uniswap swap
        - List opcodes used
        - Zk-counters used
        - Ratios
3. Next steps
    - zkGas - zkGasLimit
    - Optimize top offenders

1

Intro

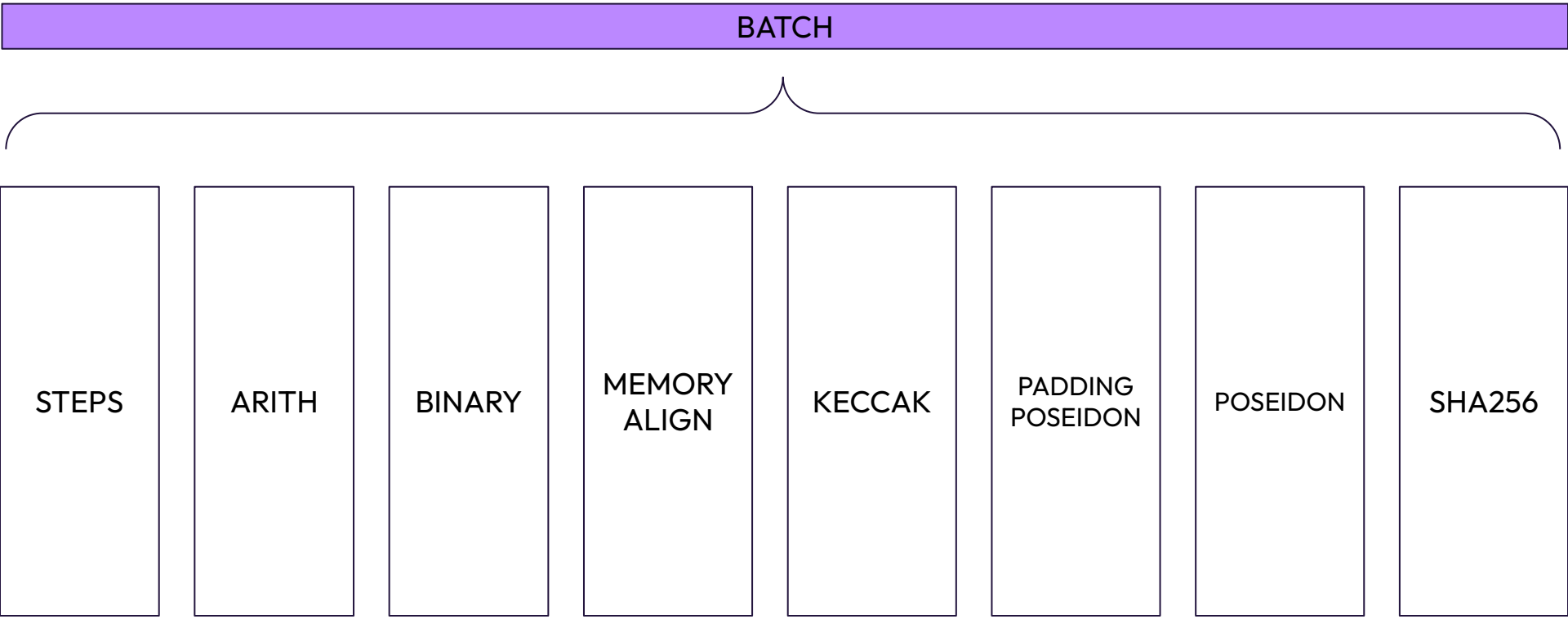
- What are zk-counters ?
  - Resources available in a proof
  - Limited for a batch
  - Multiple resources available

	Unit	**Resources**	**Measured in**	**Dimension**
Ethereum	Block	Execute computation	GAS	single
zkEVM	Batch	Proof computation	zk-counters	multiple

INTRO: zk-counters types

```
104  ; COUNTERS
105  CONST %MIN_STEPS_FINISH_BATCH = 200 ; min steps to finish tx
106  CONST %TOTAL_STEPS_LIMIT = 2**25
107
108  CONST %MAX_CNT_STEPS_LIMIT = %TOTAL_STEPS_LIMIT - %MIN_STEPS_FINISH_BATCH
109  CONST %MAX_CNT_ARITH_LIMIT = %TOTAL_STEPS_LIMIT / 32
110  CONST %MAX_CNT_BINARY_LIMIT = %TOTAL_STEPS_LIMIT / 16
111  CONST %MAX_CNT_MEM_ALIGN_LIMIT = %TOTAL_STEPS_LIMIT / 32
112  CONST %MAX_CNT_KECCAK_F_LIMIT = (%TOTAL_STEPS_LIMIT / 155286) * 44
113  CONST %MAX_CNT_PADDING_PG_LIMIT = (%TOTAL_STEPS_LIMIT / 56)
114  CONST %MAX_CNT_POSEIDON_G_LIMIT = (%TOTAL_STEPS_LIMIT / 30)
115  CONST %MAX_CNT_SHA256_F_LIMIT = ((%TOTAL_STEPS_LIMIT - 1) / 31488) * 7
116
117  CONST %SAFE_RANGE = 80 ; safe guard counters to not take into account (%RANGE = 1 / SAFE_RANGE)
118
119  CONST %MAX_CNT_STEPS = %MAX_CNT_STEPS_LIMIT - (%MAX_CNT_STEPS_LIMIT / %SAFE_RANGE)
120  CONST %MAX_CNT_ARITH = %MAX_CNT_ARITH_LIMIT - (%MAX_CNT_ARITH_LIMIT / %SAFE_RANGE)
121  CONST %MAX_CNT_BINARY = %MAX_CNT_BINARY_LIMIT - (%MAX_CNT_BINARY_LIMIT / %SAFE_RANGE)
122  CONST %MAX_CNT_MEM_ALIGN = %MAX_CNT_MEM_ALIGN_LIMIT - (%MAX_CNT_MEM_ALIGN_LIMIT / %SAFE_RANGE)
123  CONST %MAX_CNT_KECCAK_F = %MAX_CNT_KECCAK_F_LIMIT - (%MAX_CNT_KECCAK_F_LIMIT / %SAFE_RANGE)
124  CONST %MAX_CNT_PADDING_PG = %MAX_CNT_PADDING_PG_LIMIT - (%MAX_CNT_PADDING_PG_LIMIT / %SAFE_RANGE)
125  CONST %MAX_CNT_POSEIDON_G = %MAX_CNT_POSEIDON_G_LIMIT - (%MAX_CNT_POSEIDON_G_LIMIT / %SAFE_RANGE)
126  CONST %MAX_CNT_SHA256_F = %MAX_CNT_SHA256_F_LIMIT - (%MAX_CNT_SHA256_F_LIMIT / %SAFE_RANGE)
127  CONST %MAX_CNT_POSEIDON_SLOAD_SSTORE = 518
```

INTRO: zk-counters types



Available counters

- Each state-machine has its own capacity
- None of those limits can be overflow

	Steps	Arith	Binary	Mem Align	Keccak	Padding Poseidon	Poseidon	Sha256
N=25	33554432	1048576	2097152	1048576	9504	599186	1118481	7455

2

Analysis offenders

Single opcodes - precompile offenders

- **KECCAK**
  - Consumes keccak state-machine depending on the length
  - Proving cost not reflected on GAS
- **CODECOPY - CALLDATACOPY**
  - Memory expansion gas computation
  - Loop copy memory regions
- **EXTCODECOPY**
  - Load entire bytecode and hash it
  - Verify hash matches with the state-tree
  - Memory operations
- **PRECOMPILED SC**
  - Requires large usage of different state-machines to prove its execution
  - Misalignment between GAS and proof pricing

Standard EVM transactions offenders

- What is the most limiting zk-counter on standard EVM transactions ?
- Assumptions:
  - N=25
  - Full batch with 1 block with N transactions

		zk-counters % ((used/available) * 100)							
	Txs/batch	Steps	Arith	Binary	Mem align	keccak	Padding poseidon	Poseidon	Sha
ETH transfer	2078	43.9	99.9	68.3	0	63	1.1	52.6	0.0
ERC20 transfer	1092	99.7	67.2	80.5	5.6	84.9	27.5	58.7	0.0
Uniswap swap	168	99.9	23.3	60	5.7	49.6	45.7	53.1	0.0

Simplification: Top opcodes steps consumers

- What are the opcodes that uses most steps ?

Top 5 consumers	Steps consumed	Hits
PUSH1	12502	658
PUSH2	9900	495
MSTORE	9375	139
POP	6940	694
PUSH20	5480	137

Opcodes: GAS Vs Steps

- What are the opcodes that consumes steps that is not reflected on its GAS
- Compute ratio between GAS Vs Steps

Opcode	GAS	Steps	Ratio (%)
CODECOPY	15	1059	70.6
MSTORE	3	103	34.3
CALLDATACOPY	12	234	19.5
PUSH32	3	50	16.6
CALLDATALOAD	3	49	16.3

Best ratio

Opcode	GAS	Steps	Ratio (%)
DUP1	3	13	4.3
JUMP	8	18	2.25
JUMPI	10	14	1.4
EXTCODESIZE	100	44	0.44
LOG3	1756	142	0.08

3

Next steps

Static approach

- Simplest one
- Price batches individually since they are monolithic
- Batch proof always cost the same regardless of the zk-counters used
  - example: 1 batch proof costs 0.01 $
- Two actors:
  - Secuencer: pay for each batch secuenced
  - Verifier: receive secuencer fees for each batch verified

Dynamic approach

- Similar to gasLimit in Ethereum
- Batches uses dynamic resources (VADCOPS)
- Prover to allocate zk-counters depending on batch state-machine usage
- Normalize all zk-counters into one single unit measure: **zkGas**
- Two actors:
  - Secuencer: sets a **zkGasLimit** that is willing to pay for each batch
  - Verifier: Proof batch and checks enough **zkGas** has been paid

FUTURE WORK: Optimize top offenders

- After analyzing the most offender opcodes, what are the next steps ?
  - Improve compiler to add new instructions
  - Improve top opcodes with those new instructions
  - Get better ratio GAS / STEPS

More on this topic ? Join workshop tomorrow 14th at 03:00PM Classrom C

Optimize zkEVM throughput: Series II

Thanks !!

Carlos Matallana
Protocol Team Lead
carlos@polygon.technology
@KrlsMata

polygon zkEVM