EPF - Nethermind/IL-EVM

IL opcodes, stats analyzer R&D, pattern implementations
&
testing

Siddharth Vaderaa

Fellow, EPF

Here’s the timeline.

Opcodes in IL

Week 6-8
Warm up task to better understand IL-EVM and also the codebase

Stats analyzer research

Week 9-12
Literature review and cementing a strategy for the next task

Stats analyzer implementation

Week 13-16:
Core Focus

Pattern implementations

Week 17-18:
21 Patterns implemented from data gathered by the analyzer

Testing

Week 19+:
Bugs were discovered and fixed for a few opcodes

Section 1

IL implementation & code DB stats

LOGx opcodes

Warm up task

- Started the very next task required on IL/EVM (mentor's next task)
- Mentor was busy with EOF
- Task was partially completed as the mentor returned to IL/EVM work and I had to start on the upcoming task.

Code DB stats

Warm up task

- Task was to obtain to top n-grams in the codedb
- Little focus on code or performance
- Done over a weekend
- Extremely slow (100x slower than the analyzer done later)

Section 2

Research and algorithm selection for stats accumulation and analysis.

Relevant Papers

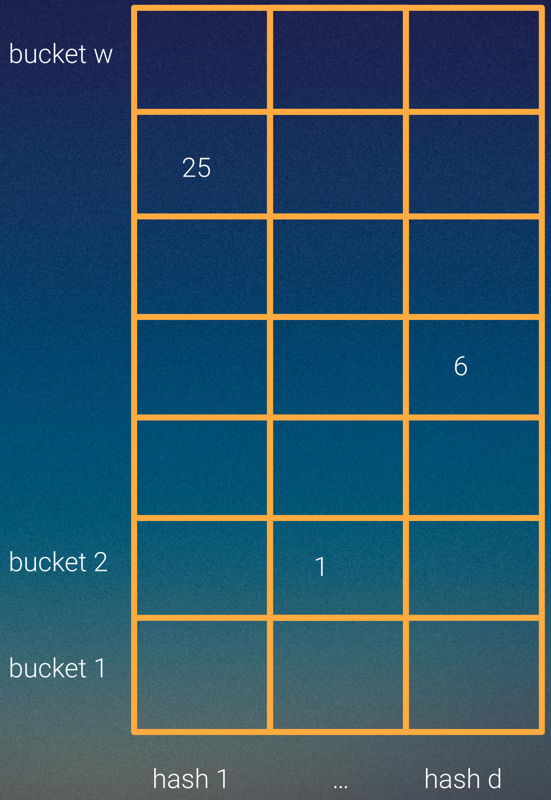
- Ben Basat, Ran et al. “Memento: Making Sliding Windows Efficient for Heavy Hitters.” *IEEE/ACM Transactions on Networking* 30 (2018): 1440-1453.
- Ben-Basat, Ran et al. “Heavy hitters in streams and sliding windows.” *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications* (2016): 1-9.
- Gou, Xiangyang et al. “Sliding Sketches: A Framework using Time Zones for Data Stream Processing in Sliding Windows.” *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020): n. Pag.
- Yang, Tong et al. “HeavyKeeper: An Accurate Algorithm for Finding Top- k Elephant Flows.” *IEEE/ACM Transactions on Networking* 27 (2019): 1845-1858.

Relevant Papers

- Li, Hua-Fu et al. “A SINGLE-SCAN ALGORITHM FOR MINING SEQUENTIAL PATTERNS FROM DATA STREAMS.” *International Journal of Innovative Computing Information and Control* 8 (2012): 1799-1820.
- Laur, Pierre-Alain et al. “Mining Sequential Patterns on Data Streams : A Near-Optimal Statistical Approach.” (2005).
- Teng, Wei-Guang et al. “A Regression-Based Temporal Pattern Mining Scheme for Data Streams.” *Very Large Data Bases Conference* (2003).
- Gou, Xiangyang et al. “Sliding Sketches: A Framework using Time Zones for Data Stream Processing in Sliding Windows.” *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020): n. Pag.
- Yang, Tong et al. “HeavyKeeper: An Accurate Algorithm for Finding Top- k Elephant Flows.” *IEEE/ACM Transactions on Networking* 27 (2019): 1845-1858.

Algorithm Selection: CM Sketch

- Cormode, Graham and S. Muthukrishnan. “An improved data stream summary: the count-min sketch and its applications.” *J. Algorithms* (2004).



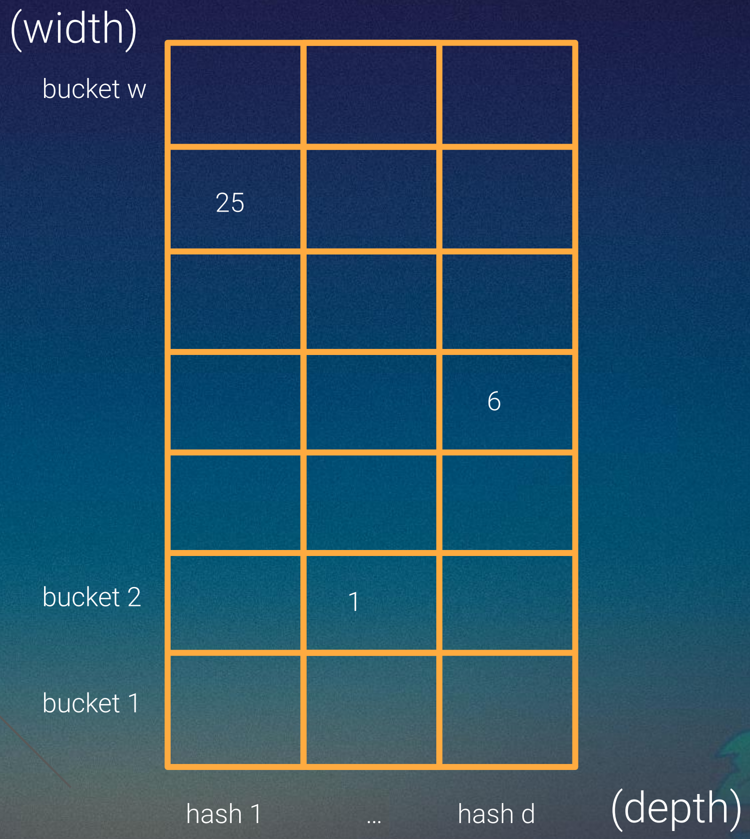
bucket w			
	25		
			6
bucket 2		1	
bucket 1			
	hash 1	…	hash d

Algorithm Selection: CM Sketch

- The error margin is maintained by controlling the counter width.
- The probability of exceeding the error margin is controlled by the number of hash functions.

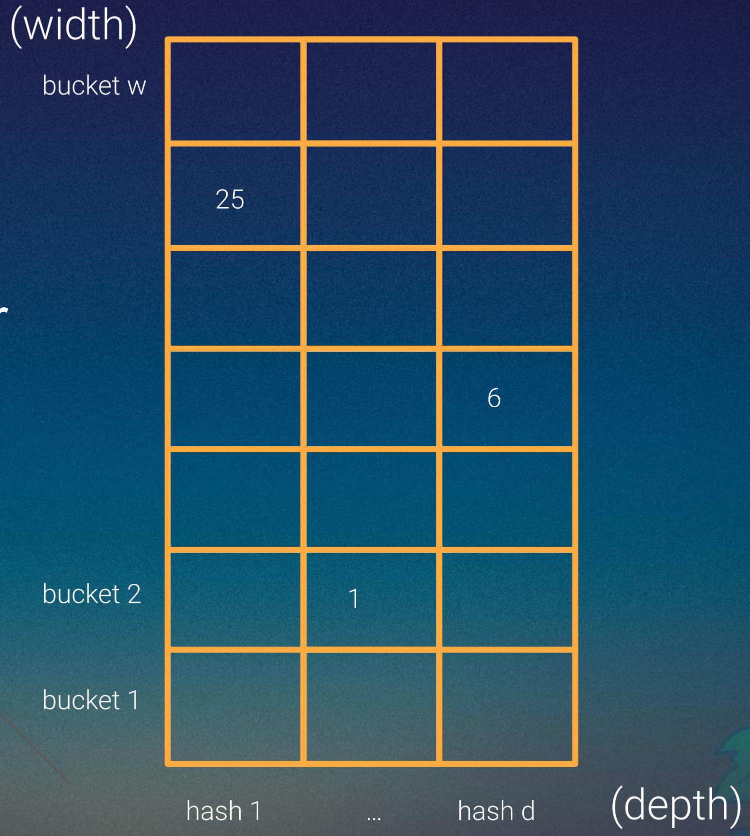
$$w = \left\lceil \frac{e}{\varepsilon} \right\rceil$$

$$d = \left\lceil \ln \frac{1}{\delta} \right\rceil$$



Algorithm Selection: CM Sketch

- The sketch component of the stats analyzer can be configured by (width,depth) or (ε,δ)
  - ○ **ϵ** controls **error bounds** through width
  - ○ **δ** controls **probability bounds** through depth.



	hash 1	...	hash d
bucket w			
	25		
			6
bucket 2		1	
bucket 1			

Section 3

Stats Analyzer

Encoding NGrams

“<<” then “ |” the opcode into a long type (8 bytes)

- Encodes Patterns of 2 to 7 Opcodes (requirement)
- Each pattern is uniquely determined by a long value

Tracking NGrams

Why track each NGram separately if we can track the biggest ngram and mask it properly?

If the current NGram of size N is greater than NGrams of size 2..N-1 (e.g. 0xffff is less than any NGram of size 3) we apply the relevant masks (eg. 0xff000000) for that size to extract and iterate over all the n-grams contained in one long value

- Iterate over the bytecode
- Encode into long
- NGram: POP, POP, ADD (long type)
- Contains 3 NGrams (3 long values)
    - POP, POP
    - POP, ADD
    - POP, POP, ADD
- STOP - resets the n-gram
- Each n-gram (long) added to sketch

Stats analyzer.

Array of sketches & a top k queue

- Iterates on bytecode.
- Encodes NGrams
- Accumulates counts in CM sketch
- New Sketch is provisioned when an error threshold is breached
- Provides the top k n-grams seen during its lifetime
- Provides the error and confidence for the stats

Tracing: Gathering execution data

Tracer plugin

- Execution data is gathered at the tx level in the EVM
- Processing kicked of after execution data is accumulated by N number of specified blocks
- Stats dumped to file after
- Does not block execution, processing done in the background
- Once the processing queue is maxed out after which the tracing turns blocking
- Done as a plugin: easy to enable when required

Output

JSON

```
"initialBlockNumber": 6748419,
"currentBlockNumber": 6751323,
"errorPerItem": 20001.678675,
"confidence": 0.96875,
"stats": [
        {
        "pattern": "PUSH2 JUMPI",
        "bytes": [
        97,
        87
        ],
        "count": 219061953
        },
        {
        "pattern": "PUSH2 JUMP",
        "bytes": [
        97,
        86
        ],
        "count": 197856326
        },

        ......
```

Plugin config

FIne grained control

- **Enabled**: Activates or Deactivates the Plugin
- **File**: Sets the file to which the stats are dumped
- **WriteFrequency**: Sets the block frequency for writing stats to disk
- **Ignore**: Sets the opcodes to ignore
- **InstructionsQueueSize**: Sets the size of the queue used to gather instructions per block
- **ProcessingQueueSize**: Sets the number of tasks that can be queued when tracing & dumping stats in background
- **SketchBuckets**: Sets the number of buckets to use in CMSketch
- **SketchHashFunctions**: Sets the number of hash functions to use in CMSketch
- **SketchMaxError**: Sets the number of buckets derived from error to use in CMSketch
- **SketchMinConfidence**: Sets the number of hash functions derived from min confidence to use in CMSketch
- **AnalyzerTopN**: Sets the number of top n-grams to track
- **AnalyzerMinSupportThreshold**: Sets the threshold for initial n-gram tracking
- **AnalyzerCapacity**: Sets the capacity of filter used for n-gram tracking
- **AnalyzerSketchBufferSize**: Sets the buffer size for sketches used by stats analyzer
- **AnalyzerSketchResetOrReuseThreshold**: Sets the threshold to reuse or provision a new sketch

Section 4

Pattern Discovery, Selection and Implementations.

Stats for mainnet and sepolia

- Data for 10,000 blocks was gathered and analyzed
- Initially 10 n-grams of size 2 were selected
- Care had to be taken that an ngram did not intersect with any ngram implemented
- Script was created to automate selection of next K n-grams based on existing patterns
- Further 11 ngrams of size 5 and larger were implemented
- Implementations were done for the pattern matching mode of IL-EVM

21 Top K Patterns implemented

Section 5

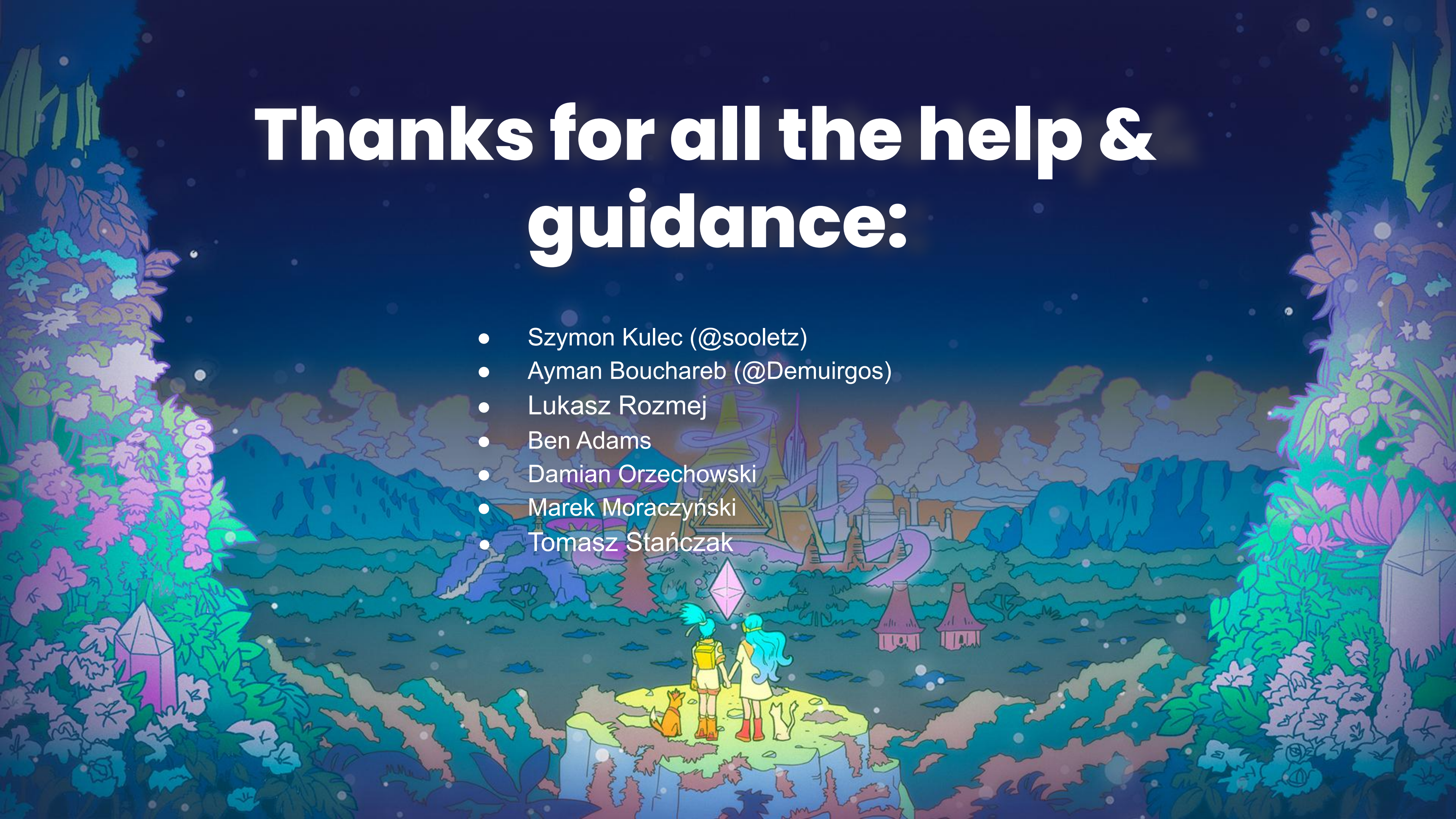
Testing

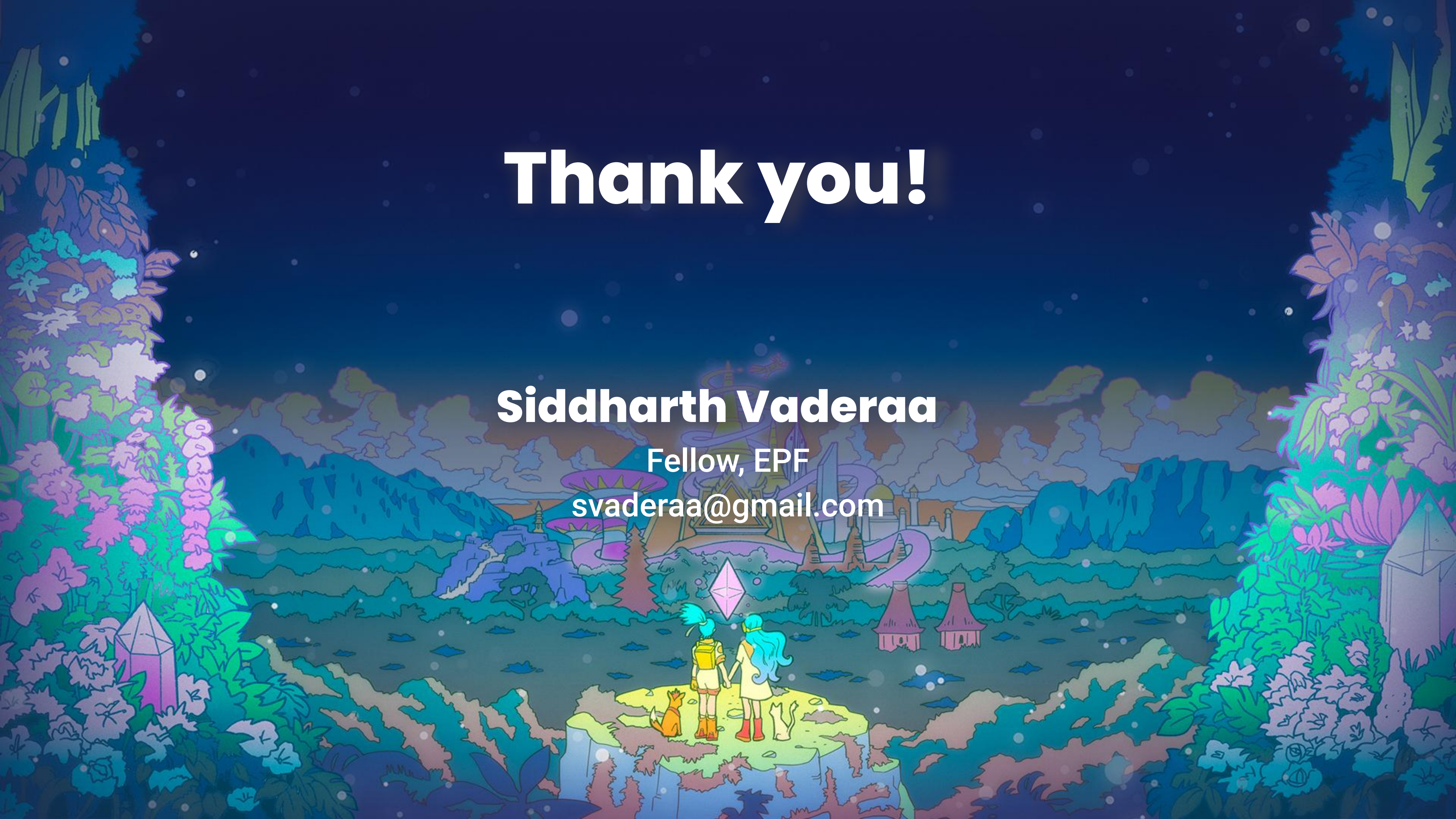
Testing (WIP)

- Node testing via syncing
- Tests for IL and ngram patterns
- Challenges encountered via state root comparison
    - Out of Gas error.
    - Spec for the opcode was not being enabled.
    - Invalid JumpDestination.
    - Contention with another pattern.
- Bug detection and fixes for multiple opcodes
- Testing for analyzer, opcodes and patterns are WIP

Thanks for all the help & guidance:

- Szymon Kulec (@sooletz)
- Ayman Bouchareb (@Demuirgos)
- Lukasz Rozmej
- Ben Adams
- Damian Orzechowski
- Marek Moraczyński
- Tomasz Stańczak





Thank you!

Siddharth Vaderaa

Fellow, EPF

svaderaa@gmail.com