I have just released a new version of Distaff VM (for more info see Introducing Distaff: a STARK-based VM written in Rust). The major changes are primarily under the hood. Specifically:

- I'm now using DEEP-FRI methodology for low-degree testing. This had significant impact on proof size (see below).
- Proof "grinding" is now supported. This allows adding extra security while keeping proof size the same (or reducing proof size for the same security) at the expense of increasing proof generation time.
- The primary field for the VM is now a 128-bit prime field. This is about 2x slower than the 64-bit field I used before (and also has higher RAM requirements) - but it comes with better soundness and also has some other future benefits.

Performance

Similar to the previous post, I benchmarked a simple Fibonacci calculator program on the new implementation. The machine is also the same: Intel Core i5-7300U @ 2.60GHz (single thread) with 8 GB of RAM. But unlike in the previous version, the proofs now target 120-bits of security.

Operation count

Execution time

Execution RAM

Verification time

Proof Size

210

260 ms

negligible

2 ms

76 KB

214

2.7 sec

160 MB

3 ms

127 KB

218

48 sec

4.4 GB

3 ms

189 KB

The parameters I used for the proof are as follows:

- Extension (or blow-up) factor: 32
- Number of queries: 50
- Grinding bits: 20

As I mentioned above, it is possible to adjust these parameters in ways that reduce proof size while increase proof time and vice-versa. The table below demonstrates this trade-off on the example of a program with 214

operations from above.

Security

3 sec

5 sec

9 sec

18 sec

100 bits

111 KB

87 KB

71 KB

65 KB

120 bits

128 KB

97 KB

84 KB

76 KB

Basically, to get a proof at 120-bit security level, you can spend 3 seconds and get a proof size of 128 KB, or you could spend 18 seconds, and the resulting proof size will be 76 KB.

A note on proof times: these times are for a relatively un-optimized implementation running in a single thread. I'd expect that for a heavily-optimized multi-threaded implementation on an 8-core processor, the 18 second proof time will go down to about 1 second.

128-bit field

As mentioned above, switching to a 128-bit field has non-negligible negative performance impact. The proof times increase by over 2x and RAM requirements also almost double.

The primary reason for switching over is that getting sufficient bits of security from a 64-bit field is non-trivial, and currently I don't know how to get 120 (or even 100) bits of security in a 64-bit field. I know that it's possible though, and it might make sense to switch back to a 64-bit field in the future.

But, 128-bit field also has a couple other benefits:

- It is much easier to construct elliptic curves over a quadratic extension of 128-bit prime fields (e.g. see Microsoft's FourQ curve). This means Distaff VM could support elliptic curve operations in the future (though, unfortunately, FourQ curve won't work).
- Since we now work with 128-bit values, the instruction set for the VM could be a bit more compact and the stack wouldn't need to grow as much for things like hashing and Merkle proof verification.

Distaff VM under the hood

I've started working on a description of how Distaff VM works under the hood. The primary description is here (you might need to refresh the page for math formulas to show up in Github). So far, it focuses on my implementation of STARKs. Any feedback is greatly appreciated!