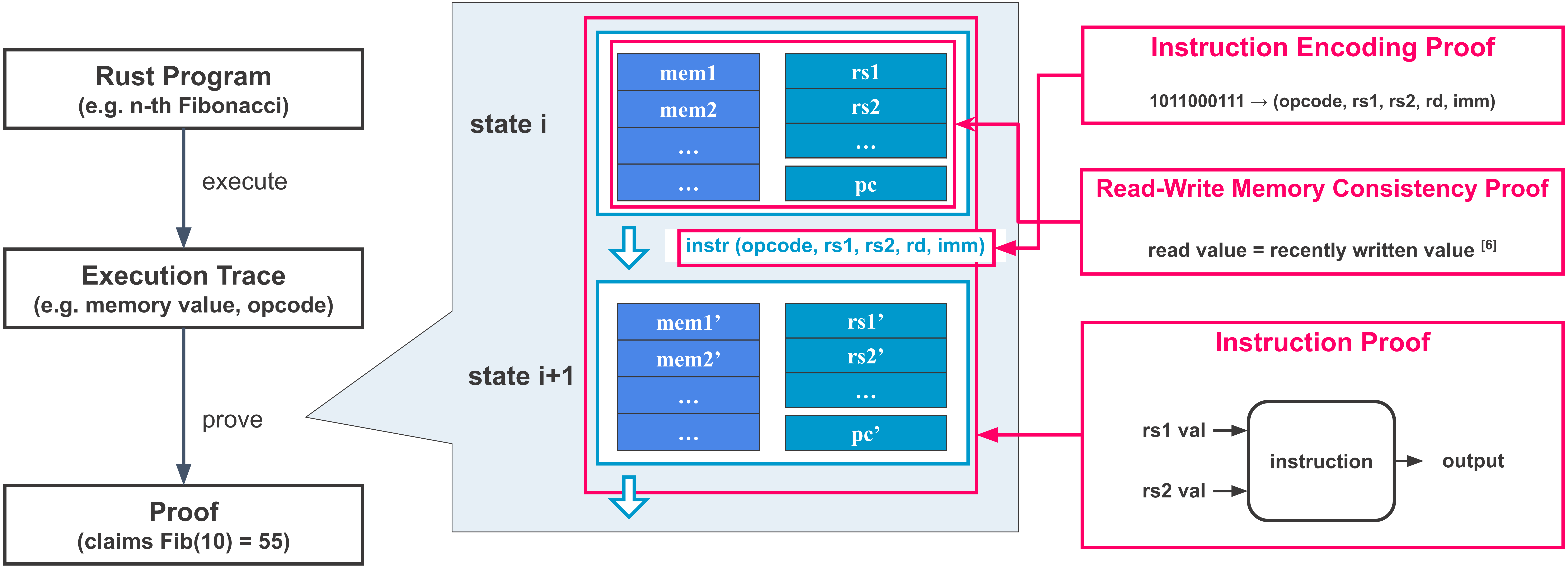
Benchmarking zkVMs: Efficiency, Bottlenecks, and Best Practices

Masato Tsutsumi, Kazue Sako

Waseda University

zkVM Concept: SNARKs for Virtual Machines

zkVM proves the correct execution of computer programs [6].

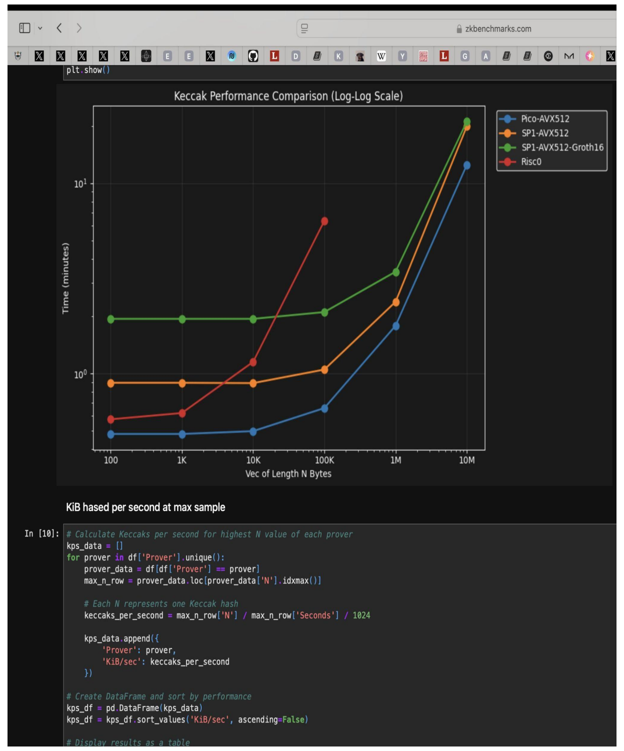


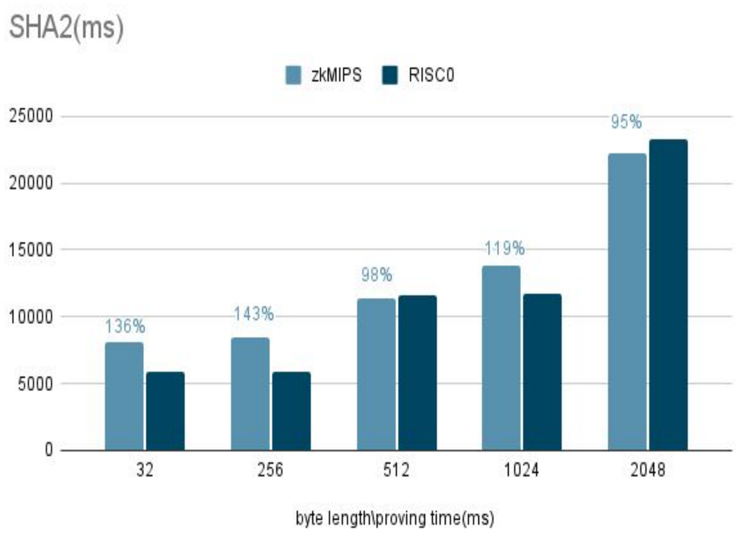
[6] Arun, A., Setty, S., Thaler, J.: Jolt: Snarks for virtual machines via lookups. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 3–33. Springer Nature Switzerland, Cham (2024)

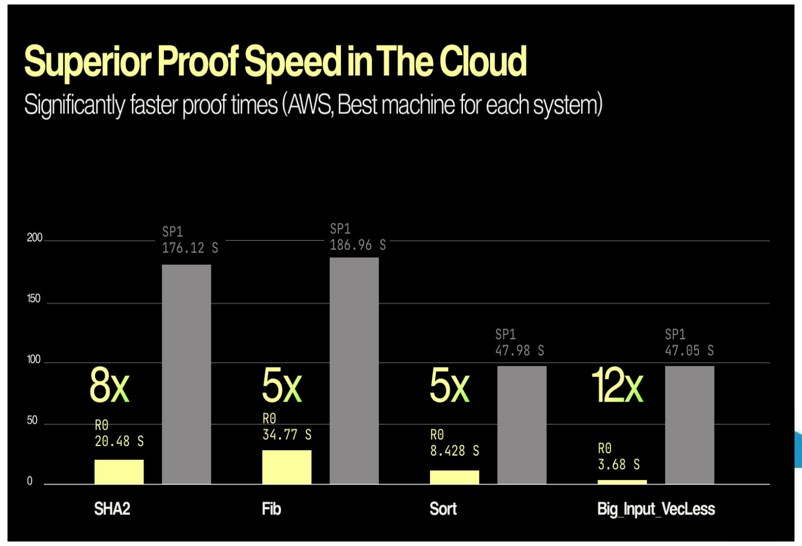
Background

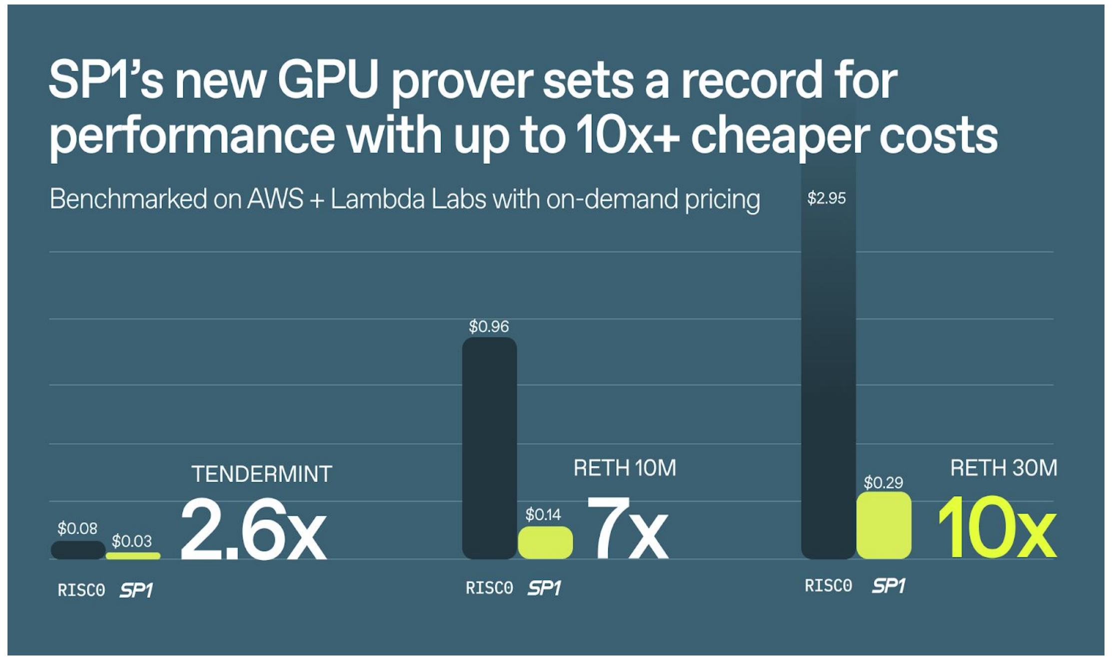
many zkVM projects claim to be the "fastest," but they are actually measured under different conditions, making it difficult to compare objectively…

actually
fast, but enormous memory consumption
fast, but only works with specific programs









https://x.com/fede_intern/status/1898018509705105529
https://x.com/RiscZero/status/1802748528168440263
https://www.zkm.io/blog/zkmips-beta-a-competitive-performance-report
https://blog.succinct.xyz/sp1-benchmarks-8-6-24/

Research Overview

When releasing new LLM model,
they cite benchmark scores as evidence of performance

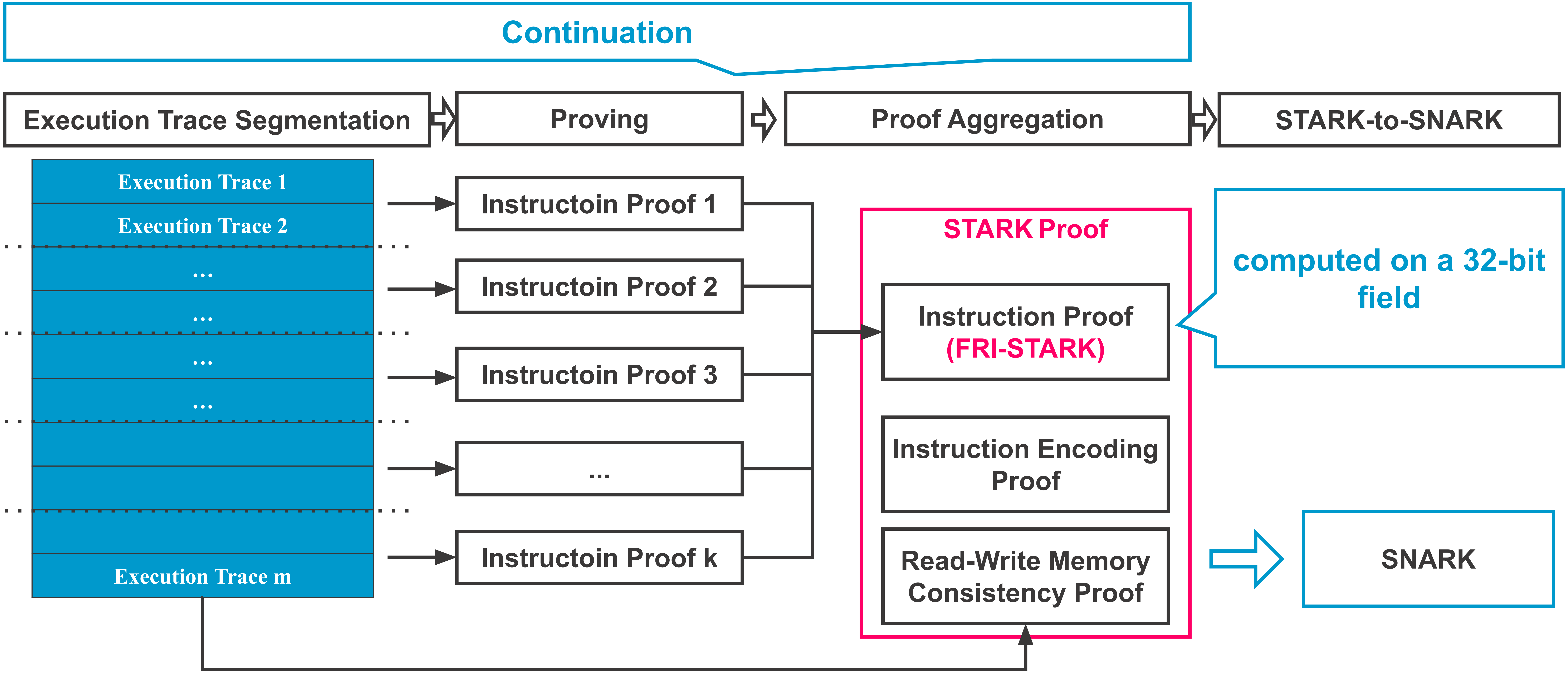
So we built a zkVM benchmark framework, and …

1. **We compared (at most) 7 zkVM projects by 4 evaluations with (at most) 4 programs.**
2. **We identified the CPU/Memory bottlenecks for each project.**
3. **We estimated the best practices for leveraging existing zkVMs.**

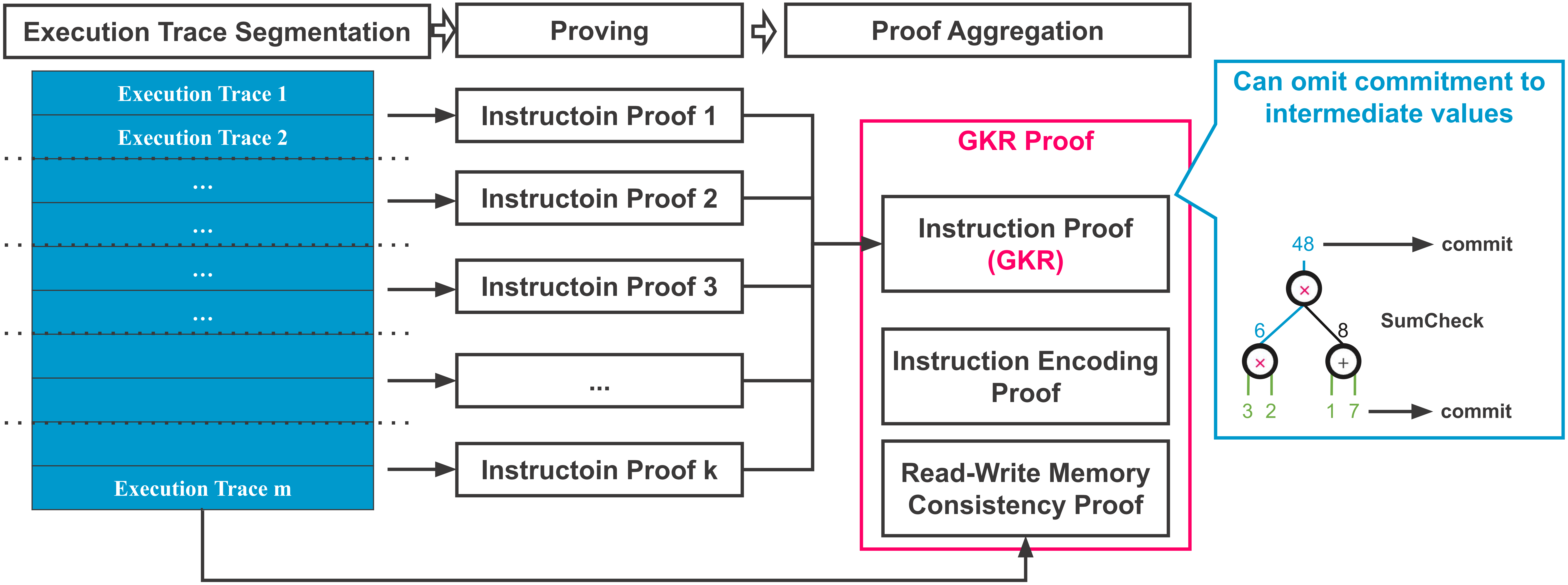
Compared Projects

- SP1, RISC Zero, Jolt (for both CPU and GPU)
- OpenVM, ZKM, Nexus, Novanet (for only CPU)
- Ceno (for only Parallel Performance)

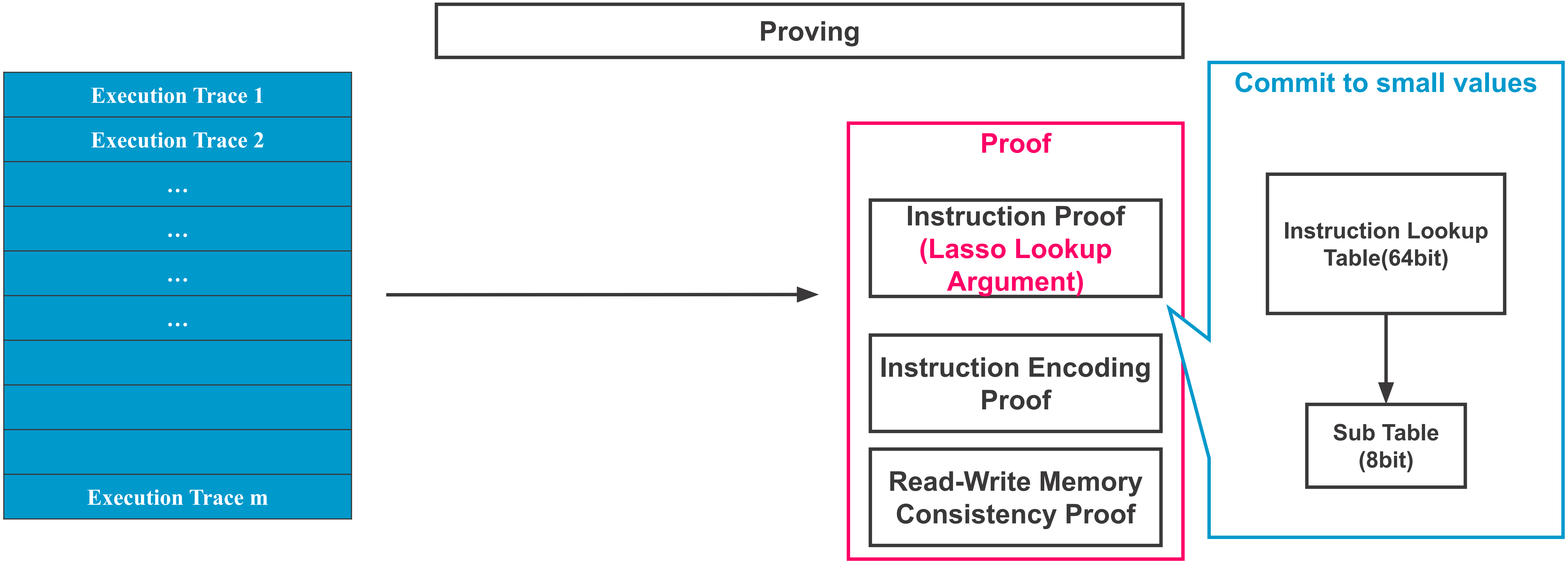
zkVM methods: SP1, RISC Zero, OpenVM, ZKM



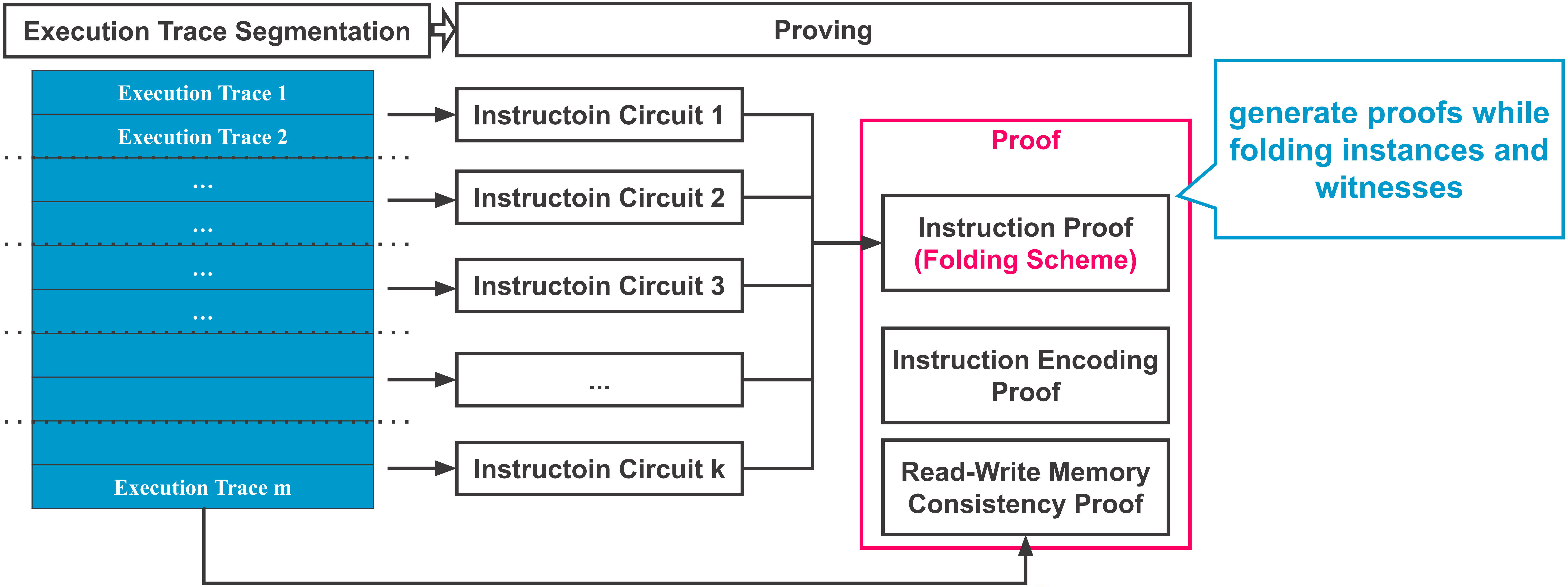
zkVM methods: Ceno



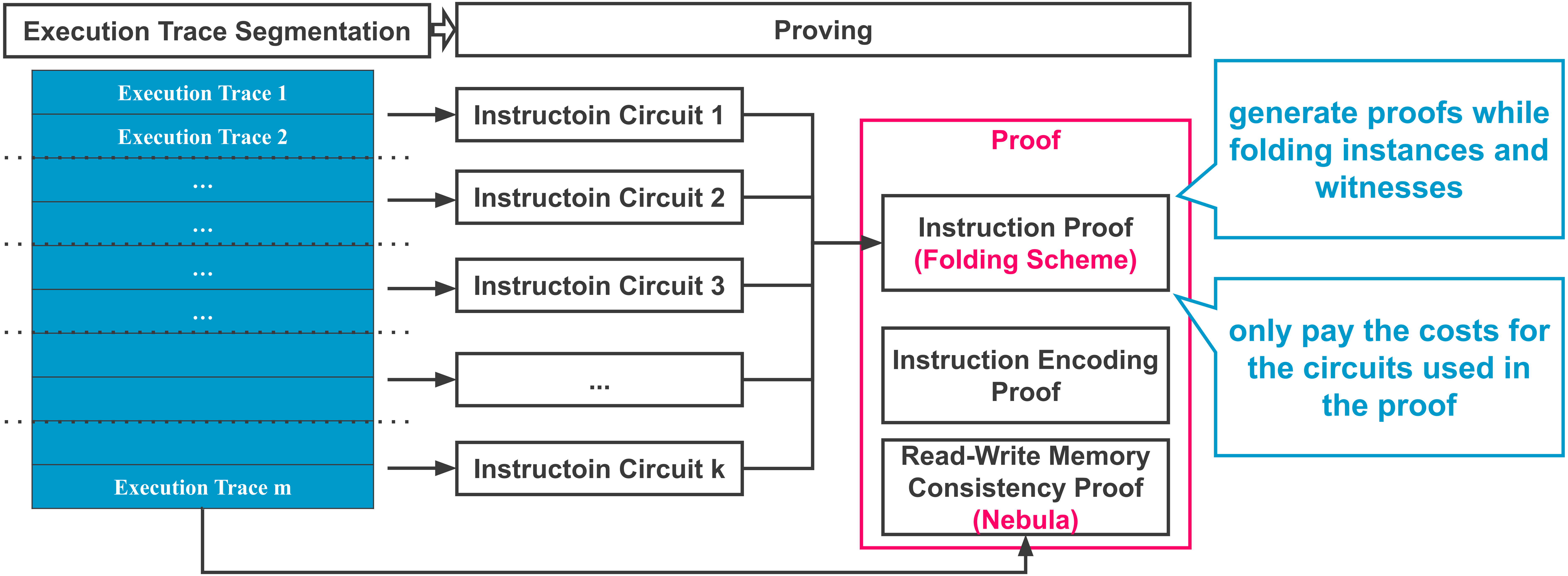
zkVM methods: Jolt



zkVM methods: Nexus



zkVM methods: Novanet



zkVM and Techniques

Project	Feature	Continuation	Precompiles
SP1	FRI-STARK	◯	SHA2, keccak, ed25519, bigint, weierstrass
RISC Zero	FRI-STARK	◯	SHA2, keccak, secp256k1, secp256r1, ed25519, RSA, bn254, bls12-381, bigint
OpenVM	FRI-STARK	◯	SHA2, keccak, ECDSA, secp256k1, secp256r1, weierstrass, msm, group, bigint, …
ZKM	FRI-STARK	◯	SHA2, keccak, bigint
Ceno	GKR	◯	SHA2, keccak, secp256k1, bn254
Jolt	Lasso Lookup Argument	✕	✕
Nexus 1.0	Nova	◯	✕
Novanet	Nebula-Nova	◯	✕

Outline

- Comparisons
- CPU and Memory Bottlenecks
- Best Practices
- Future Challenges

Comparison Setting

Test Environment

AWS EC2 g5.16xlarge
- 64 vCPUs
- 256GB RAM
- 24GB GPU Memory

Compared Projects

- SP1 (CPU/GPU)
- RISC Zero (CPU/GPU)
- ZKM
- Novanet
- OpenVM
- Jolt (CPU/GPU)
- Nexus

Test Programs

- 100k-th Fibonacci
- SHA2-2048
- ECDSA Verification on secp256k1
- EVM Execution (100 simple eth_transfer)

Evaluations

- Time Efficiency
  - prover time
  - proven less than one minute
  - changing input
  - changing number of cores
- Memory Efficiency

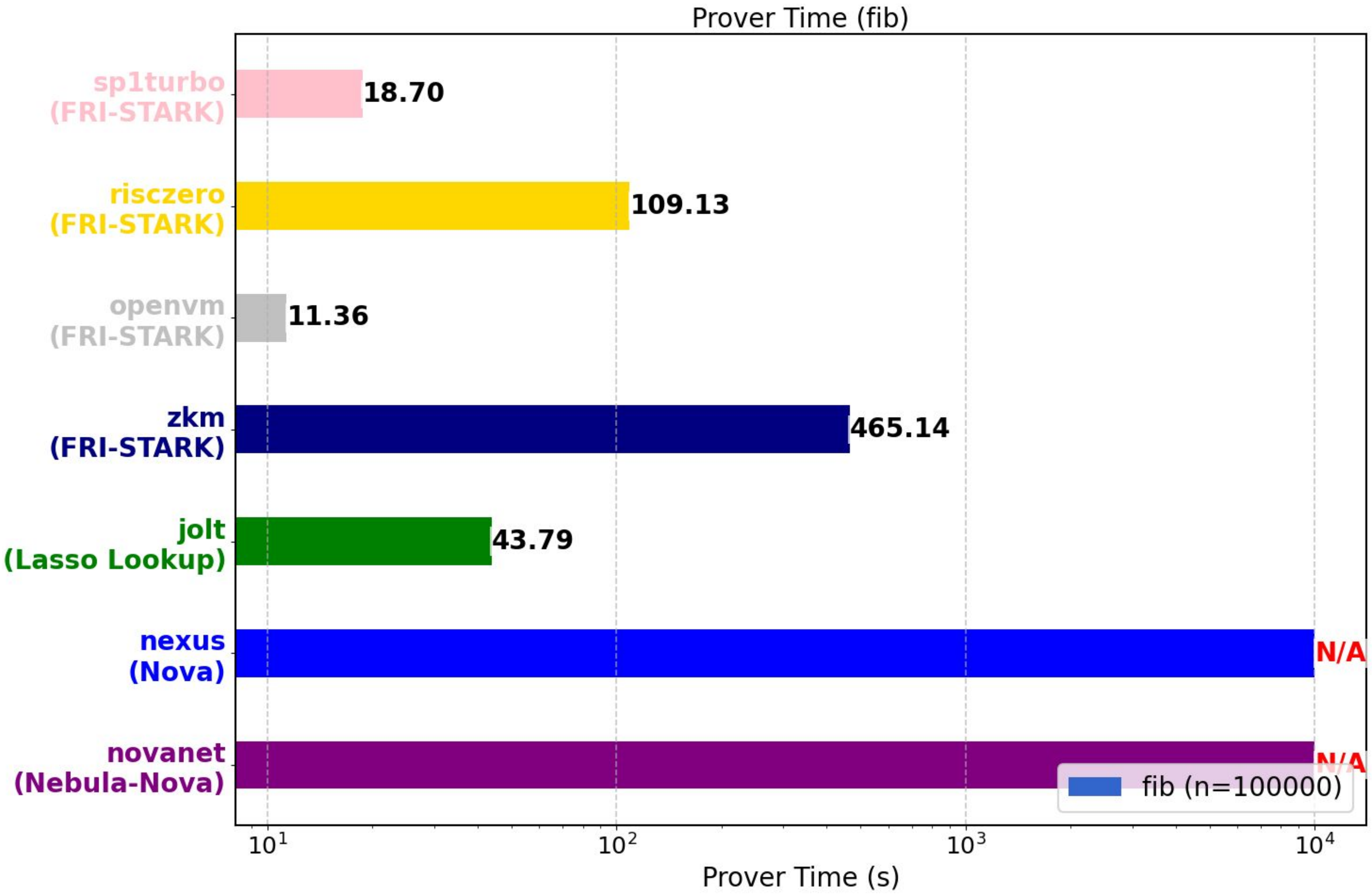
Available on:
https://github.com/grandchildrice/zkvm-benchmarks

Program and Cycle Counts

Project	100k-th Fibonacci	SHA2-2048	ECDSA Verification (k256)	EVM Execution (100 eth_transfer txs)
SP1	**609k**	**4.4M**	**4.4M**	**5.2M**
RISC Zero	**2.4M**	**262k**	**226k**	**5.1M**
OpenVM	not reported			
ZKM	**2.7M**	**166k**	**7.4M**	**5.6M**
Ceno	not reported			
Jolt	**2.4M**	**200k**	**5.5M**	**9.4M**
Nexus 1.0	not reported			
Novanet	not reported			

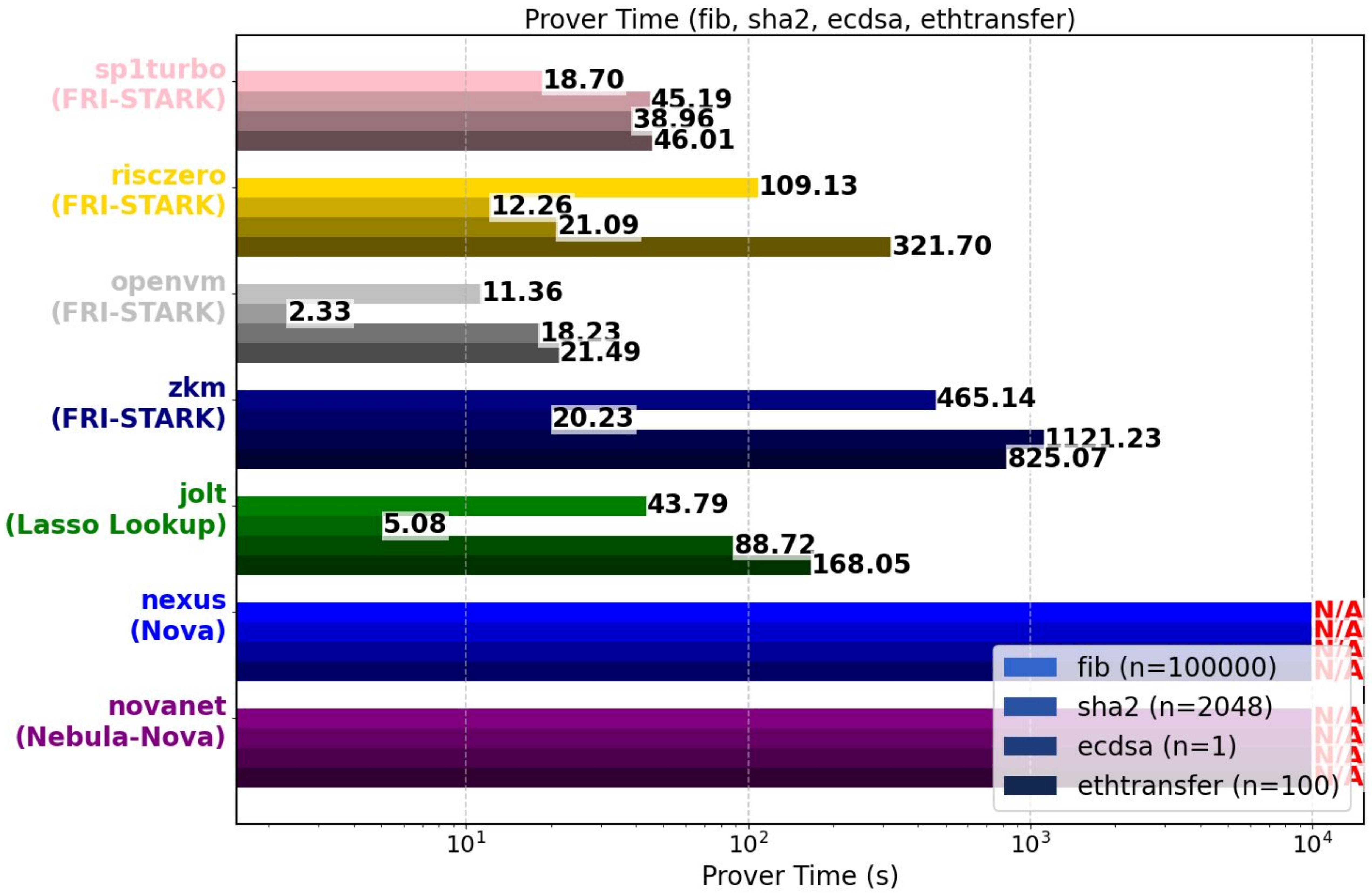
Prover Time (s) for 100k-th Fibonacci by CPU

- OpenVM, SP1, Jolt are efficient



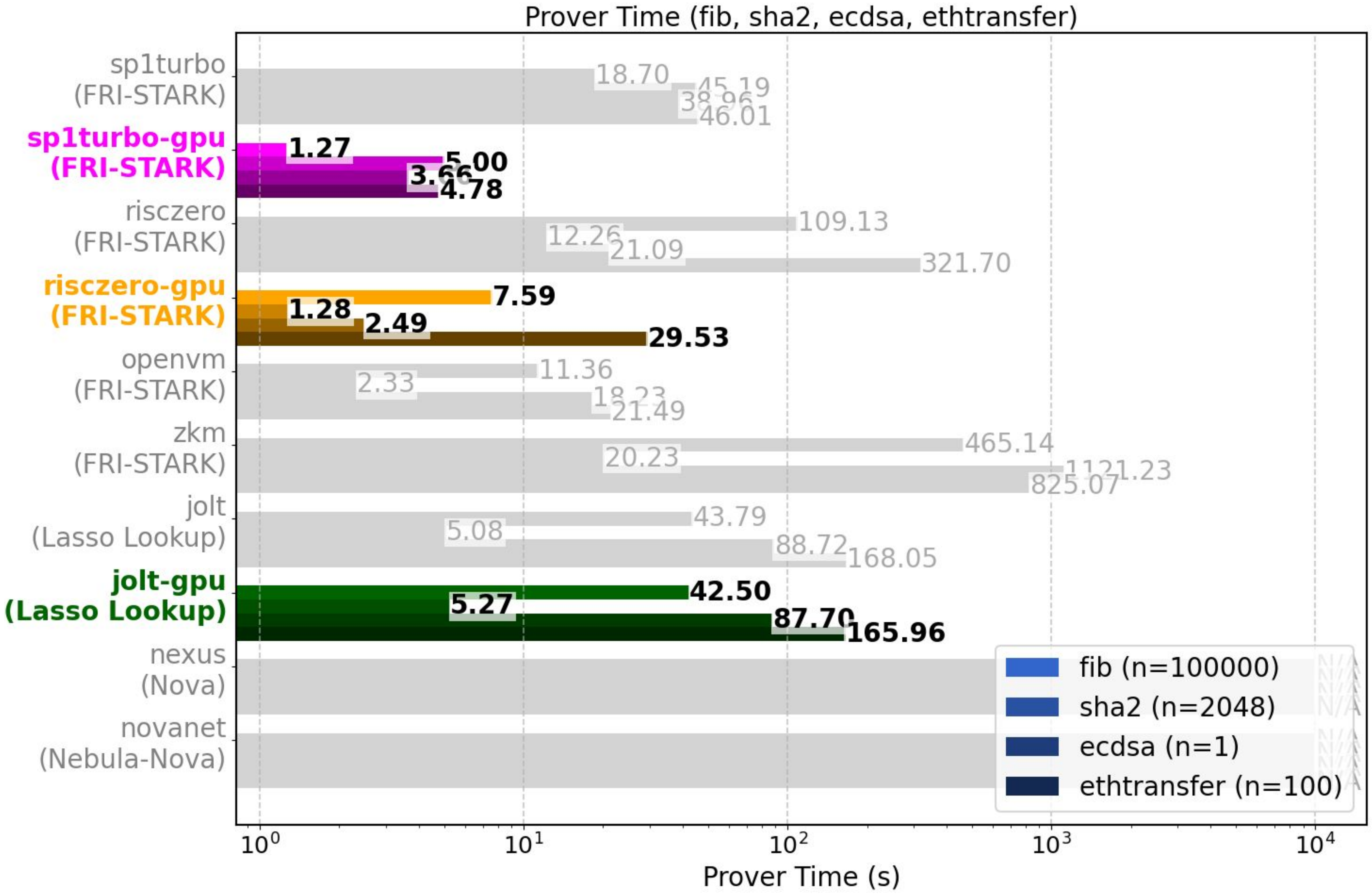
Prover Time (s) for 100k-th Fibonacci, SHA2-2048, k256-ECDSA, 100 EVM Execution by CPU

- even with close cycle counts, there is a difference in prover time
- OpenVM: fastest in all
- Jolt: second fastest in SHA2



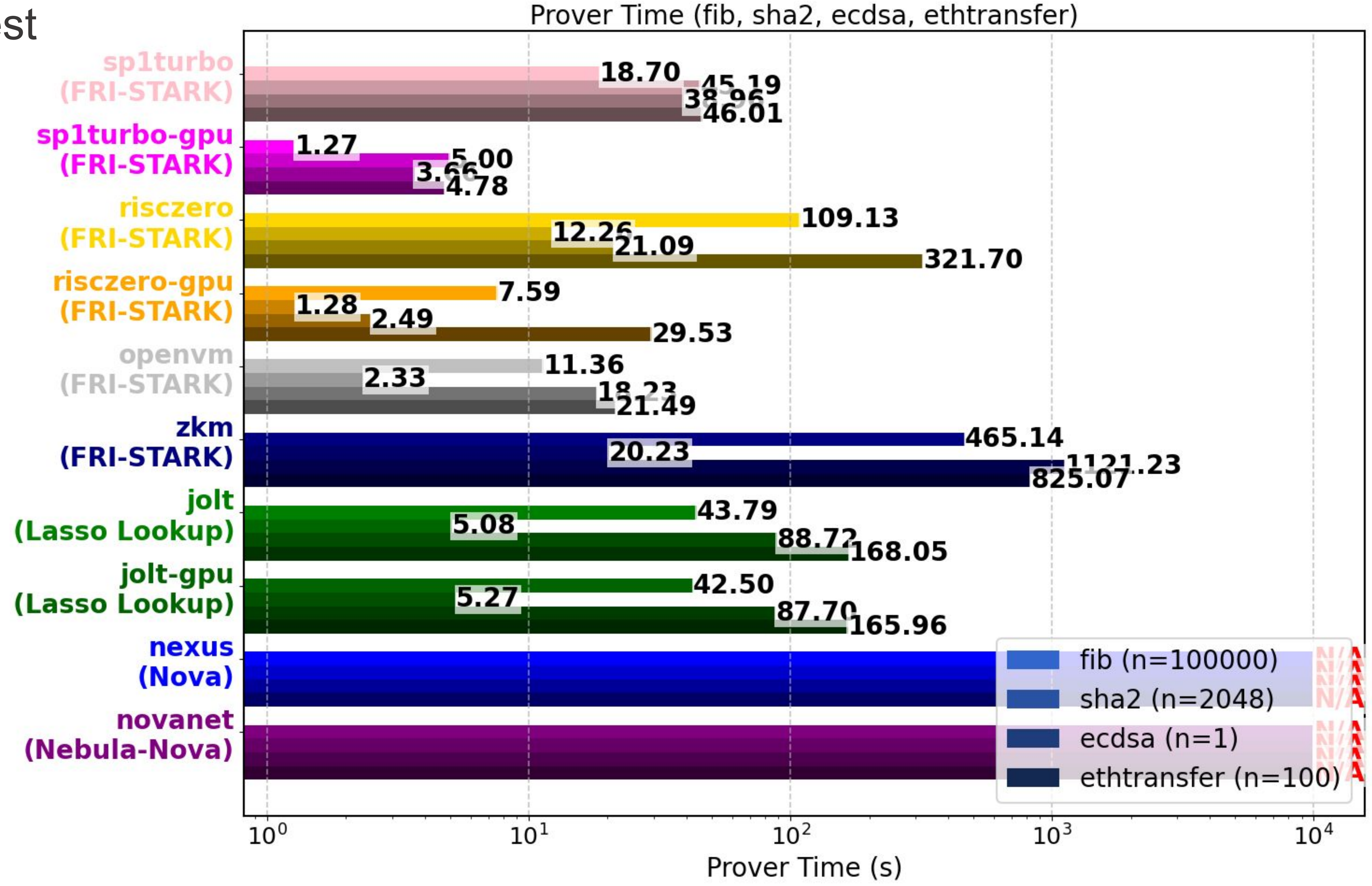
Prover Time (s) for 100k-th Fibonacci, SHA2-2048, k256-ECDSA, 100 EVM Execution by GPU

- SP1/RISC Zero: much more efficient than CPU
- Jolt: little difference, as only supports GPU acceleration for some components
- SP1: 4.7s on 100 ethtransfers, achieving the 12s real time proving is promising for practical applications
- more complex contract executions needed to be compared though



Time Efficiency

- SP1's GPU was the fastest
- followed by
  RISC Zero's GPU
  and OpenVM's CPU



Progress since submission of the paper

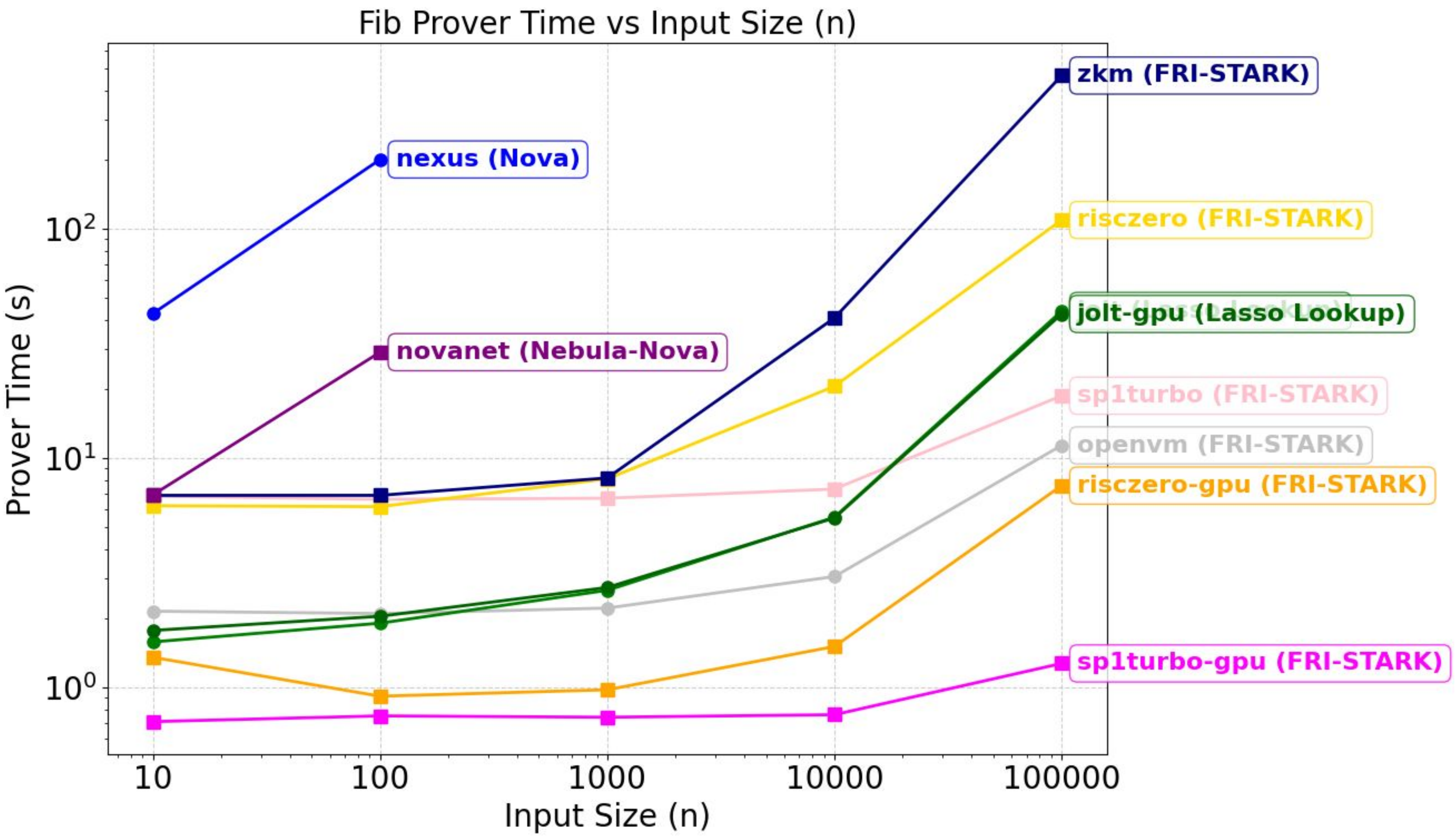
	On paper	Now
Projects	• SP1 • RISC Zero • Jolt • Ceno • Nexus	• SP1 • RISC Zero • OpenVM • ZKM • Jolt • Nexus • Novanet
Programs	• Fibonacci • MatrixOps	• Fibonacci • SHA2 • ECDSA Verification • EVM Execution

ClankPan from Novanet contributed so much

Table for whether proven in less than one minute

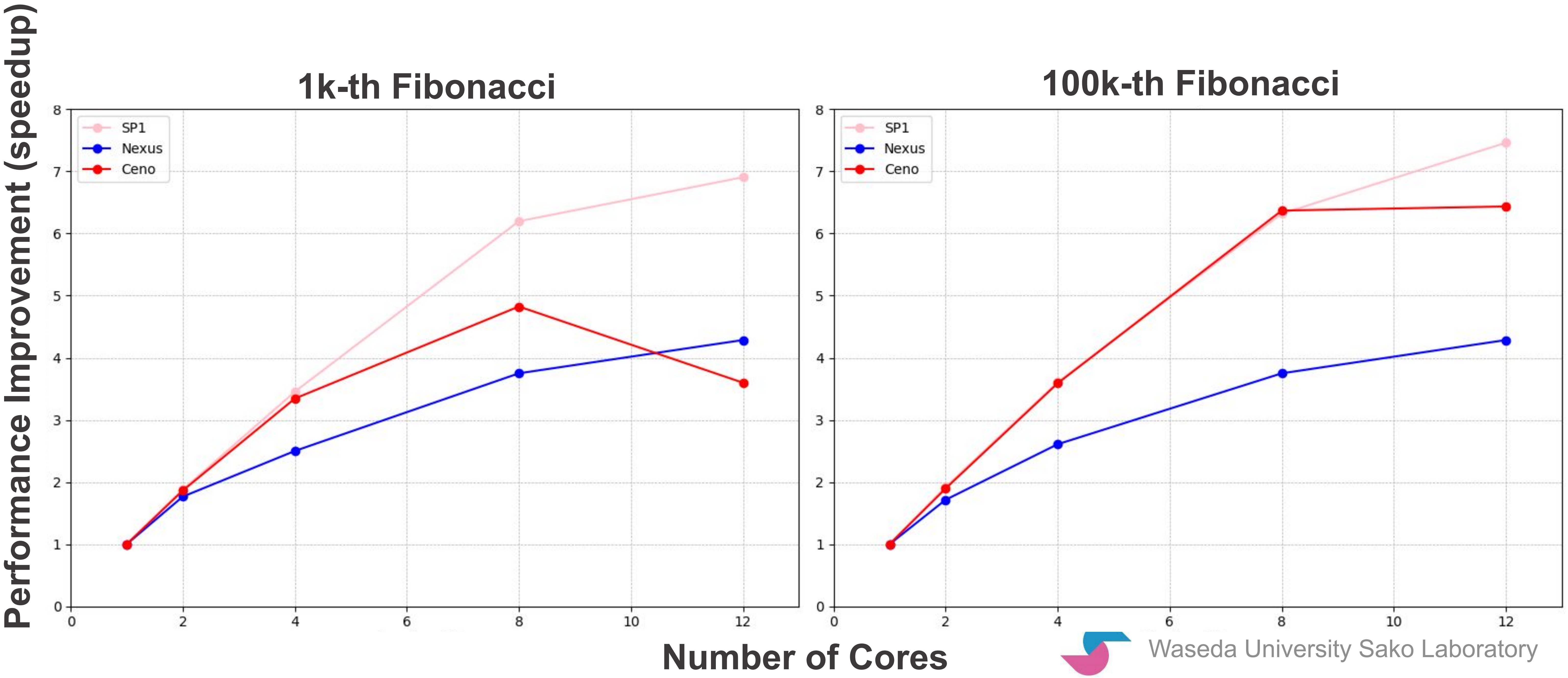
	100k-th Fibonacci	SHA2-2048	ECDSA-Verify	EVM Execution 100 ETHTransfer	Generalizability
SP1	✅	✅	✅	✅	◯
RISC Zero	✅	✅	✅	✅	◯
ZKM	❌	✅	❌	❌	△
Novanet	❌	❌	❌	❌	×
OpenVM	✅	✅	✅	✅	◯
Jolt	✅	✅	❌	❌	△
Nexus	❌	❌	❌	❌	×

Prover Time increase when changing Fibonacci's input n



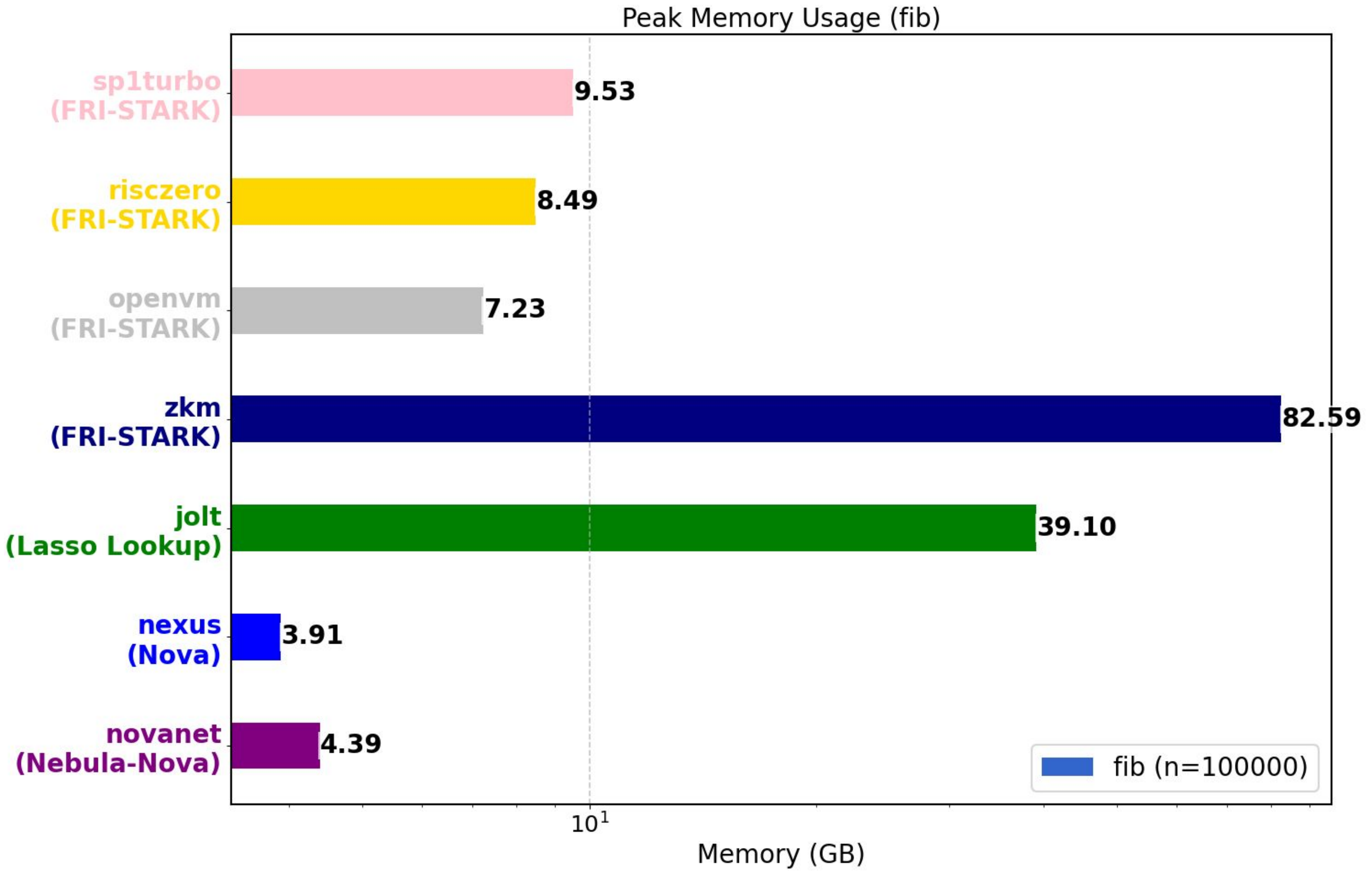
Performance improvement rate of Prover Time from single core when changing the number of CPU cores

- Here 8 cores appear to have the best performance when viewed per core.



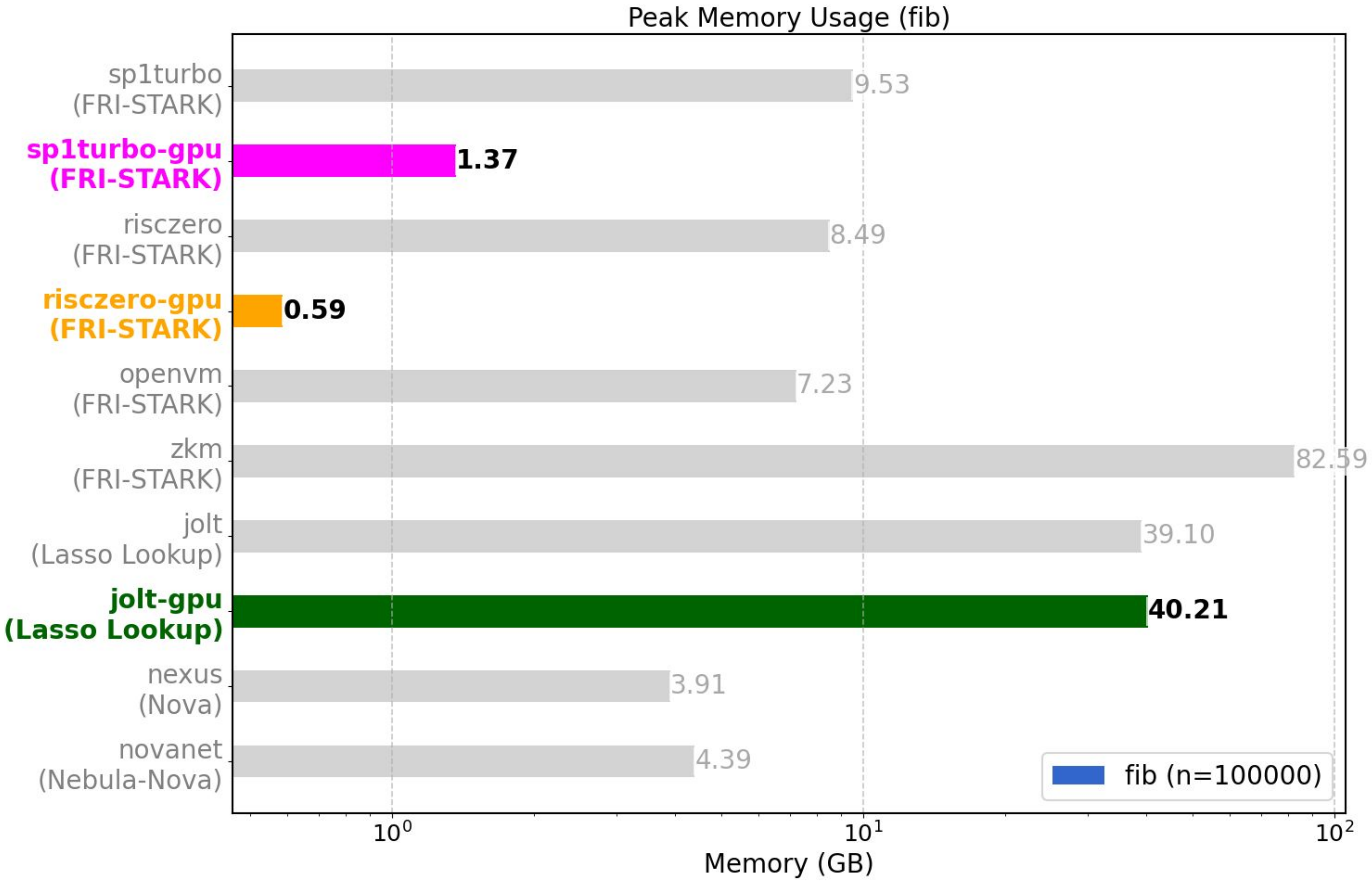
Peak Memory Usage (GB) for 100k-th Fibonacci by CPU

- Nexus, Novanet: constant event with increasing input sizes
- However, they trade this for significantly lower time efficiency

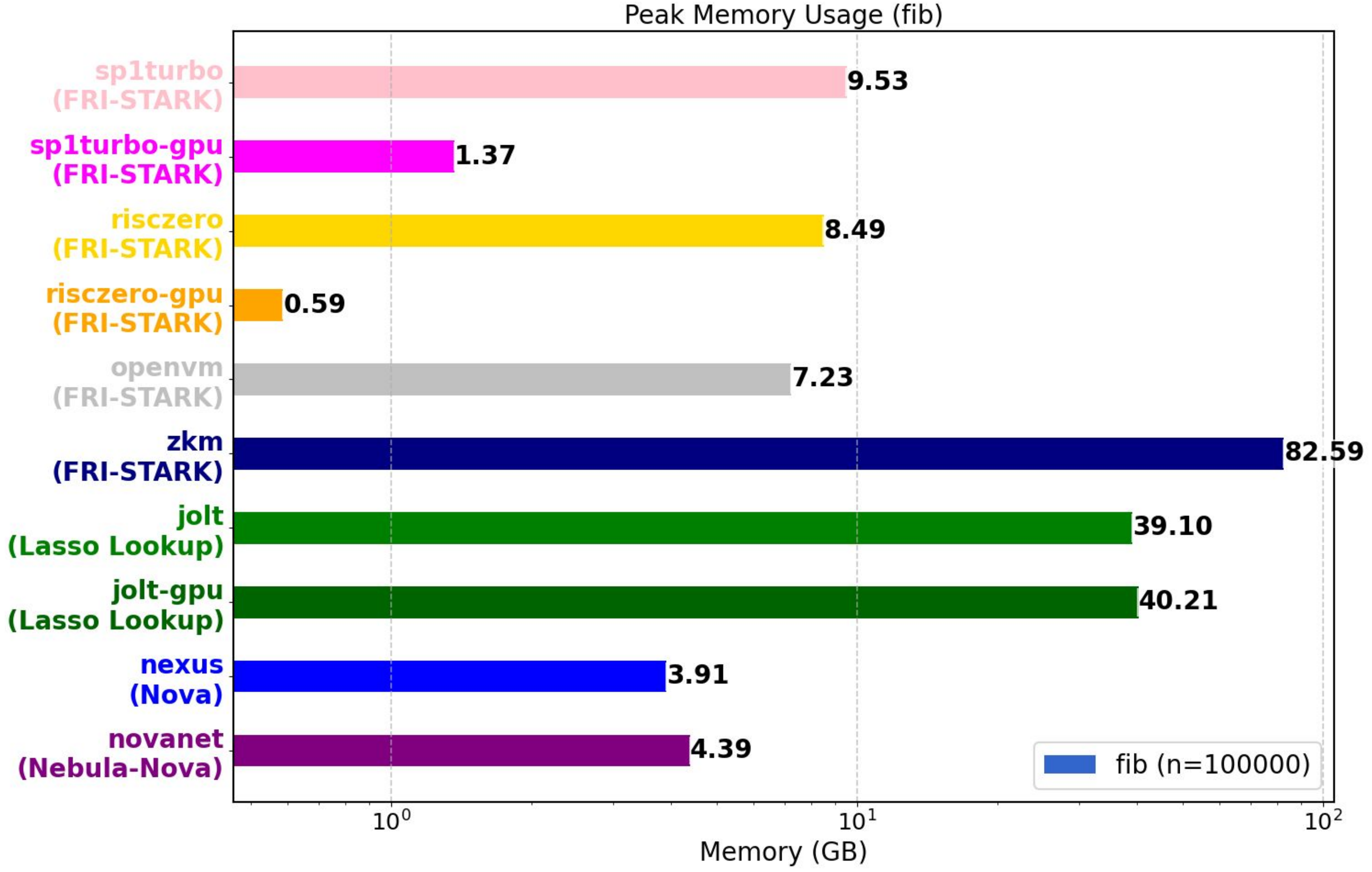


Peak Memory Usage (GB) for 100k-th Fibonacci by GPU

- SP1/RISC Zero: much lower than CPUs
- However, they used around 24GB of GPU memory instead
- Jolt: No difference between CPU and GPU



Peak Memory Usage (GB) for 100k-th Fibonacci by CPU and GPU



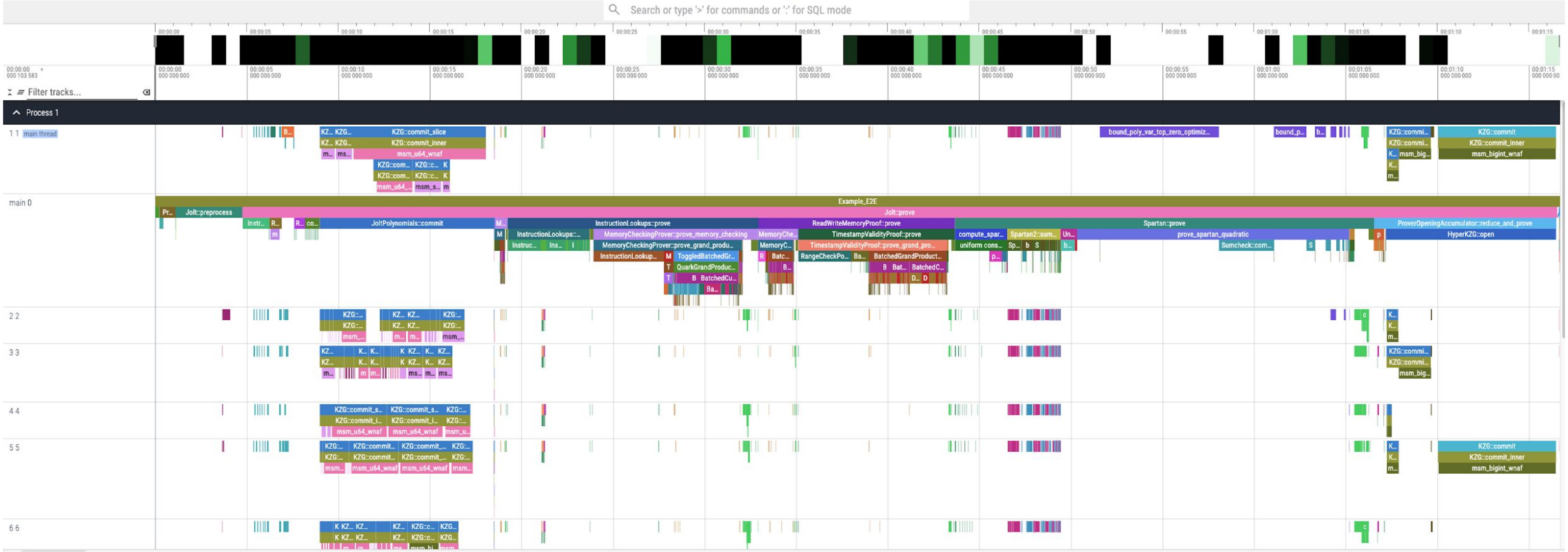
on ECDSA Verification

Nexus/Novanet: 5GB
ZKM/Jolt: ~110GB
Others: 30~50GB

Analyzing CPU/Memory Bottlenecks

	CPU Bottlenecks	Memory Bottlenecks
Jolt	Lasso’s SumCheck protocol, Spartan	Polynomial generation in Lasso
RISC Zero, SP1, ZKM, OpenVM	Proof aggregation, Commitment generation	Merkle tree construction
Ceno	SumCheck protocol in GKR	SumCheck protocol in GKR
Nexus, Novanet	Folding of instance-witness pairs	Folding of instance-witness pairs

Analyzing CPU/Memory Bottlenecks



Best Practices for leveraging existing zkVMs.

1. **zkVM Selection Criteria**
   a. SP1, RISC Zero, OpenVM: seemingly good for general-purpose use cases
   b. should choose a project that has the instructions you use as a precompile
   c. recommend measuring the best number of cores.
2. **GPU Specifications**
   a. 24GB GPU Memory required
3. **Memory Specifications**
   a. General: 32GB RAM or more recommended
   b. Jolt, ZKM: 128GB RAM or more recommended

Summary

So we built a zkVM benchmark framework, and …

1. **We compared** (at most) **7 zkVM projects by 4 evaluations with** (at most) **4 programs.**
2. **We identified the CPU/Memory bottlenecks for each project.**
3. **We estimated the best practices for leveraging existing zkVMs.**

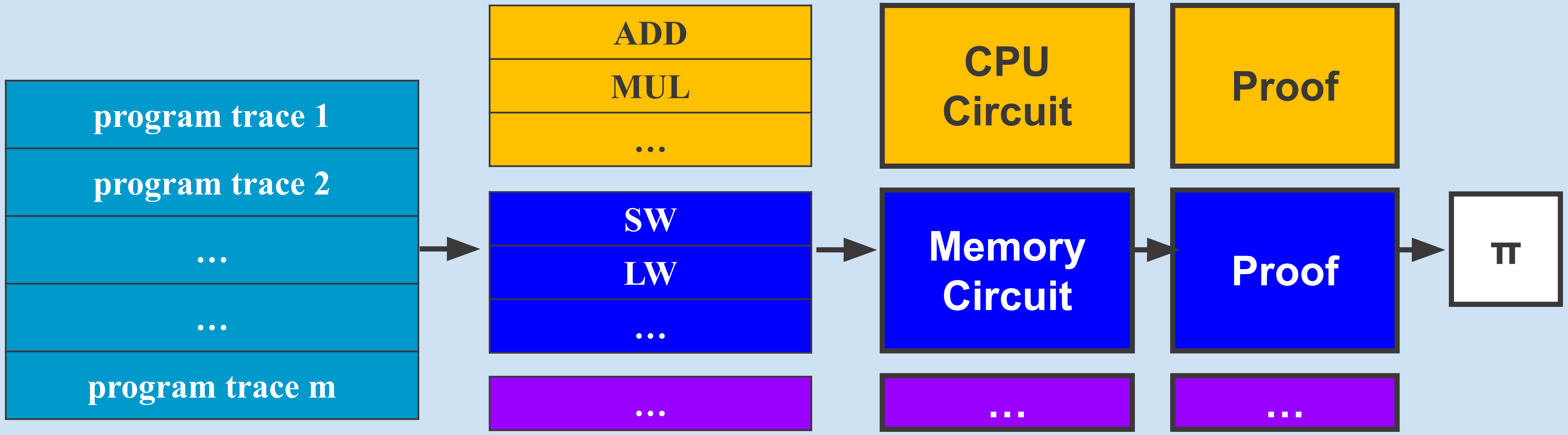
- Future work: conduct more detailed bottleneck analysis, support additional zkVM projects, expand more programs, incorporate more performance metrics, enhance parallel performance analysis

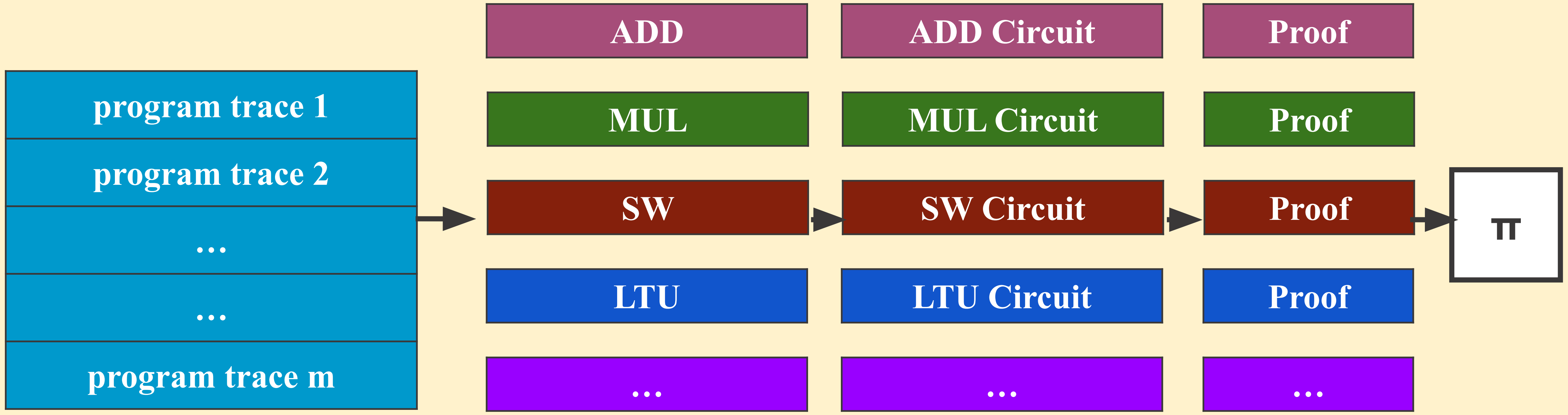
Thank you

https://github.com/grandchildrice/zkvm-benchmarks

Common zkVM Architecture

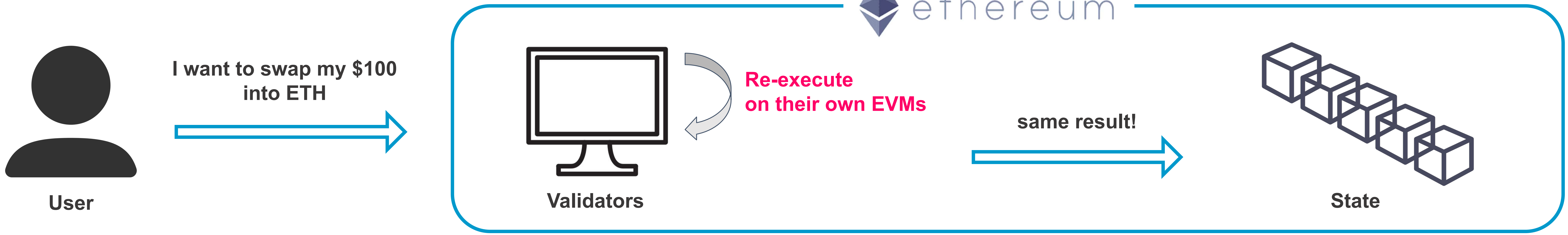
Project	Proof System	Architecture
SP1	FRI-STARK	**vRAM Style: executes custom circuits grouped by each instruction type**
RISC Zero	FRI-STARK	
ZKM	FRI-STARK	
Novanet	Nebula-Nova	
Ceno	GKR	
OpenVM	FRI-STARK	**Modular Style: executes custom circuits defined for each opcode type**
Jolt	Lasso Lookup	
Nexus 1.0	Nova	
Valida	FRI-STARK	





zkVM use case: Scaling Ethereum Smart Contracts

Ethereum Today



Ethereum with zkVMs