Simulation analysis

`leios-2025w26`

Three experiments

1. Unsharded (“simplest”) variant of Leios, with various levels of conflicting transactions
   https://github.com/input-output-hk/ouroboros-leios/blob/main/analysis/sims/2025w26/analysis-simple-conflicts.ipynb
2. Nine variants of base protocol vs sharing
   https://github.com/input-output-hk/ouroboros-leios/blob/main/analysis/sims/2025w26/analysis-variants-sharding.ipynb
3. Praos with large blocks

   https://github.com/input-output-hk/ouroboros-leios/blob/main/analysis/sims/2025w26/analysis-praos.ipynb

Nine Leios variants

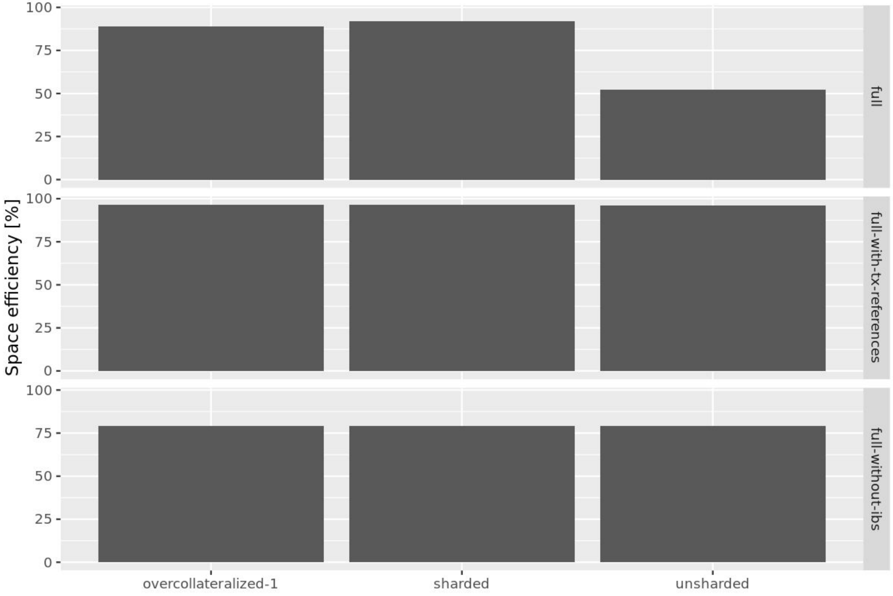
Nine realizations of Leios: 3 variants ✕ 3 shard strategies

Variant	Sharding	IB per TX	TX per IB	IB per EB	EB per IB	Spatial Efficiency	Time to IB	Time to EB	Time to RB
full	unsharded	1.896	617.043	5.172	11.022	52.307%	8.510s	29.806s	64.404s
full	sharded	1.071	46.191	37.200	9.911	92.043%	80.094s	101.019s	137.362s
full	overcollateralized 1x	1.106	47.706	37.200	9.911	89.097%	37.031s	57.882s	94.155s
full-without-ibs	unsharded	-	-	-	-	79.050%	-	4.779s	43.057s
full-without-ibs	sharded	-	-	-	-	79.052%	-	4.784s	43.052s
full-without-ibs	overcollateralized 1x	-	-	-	-	79.053%	-	4.783s	43.053s
full-with-tx-references	unsharded	1.704	554.638	5.172	11.022	95.999%	7.409s	28.882s	63.763s
full-with-tx-references	sharded	1.071	46.191	37.200	9.911	96.466%	80.058s	100.983s	137.324s
full-with-tx-references	overcollateralized 1x	1.102	47.522	37.200	9.911	96.413%	36.972s	57.826s	94.112s

Spatial efficiency

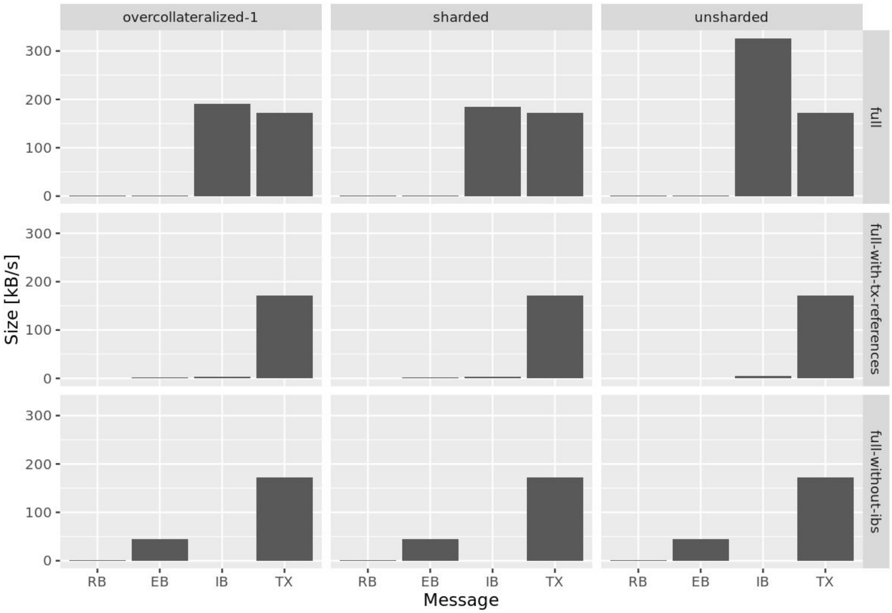
Spatial efficiency (size of txs on ledger / size of non-tx persisted data)

Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage



Size of persisted data

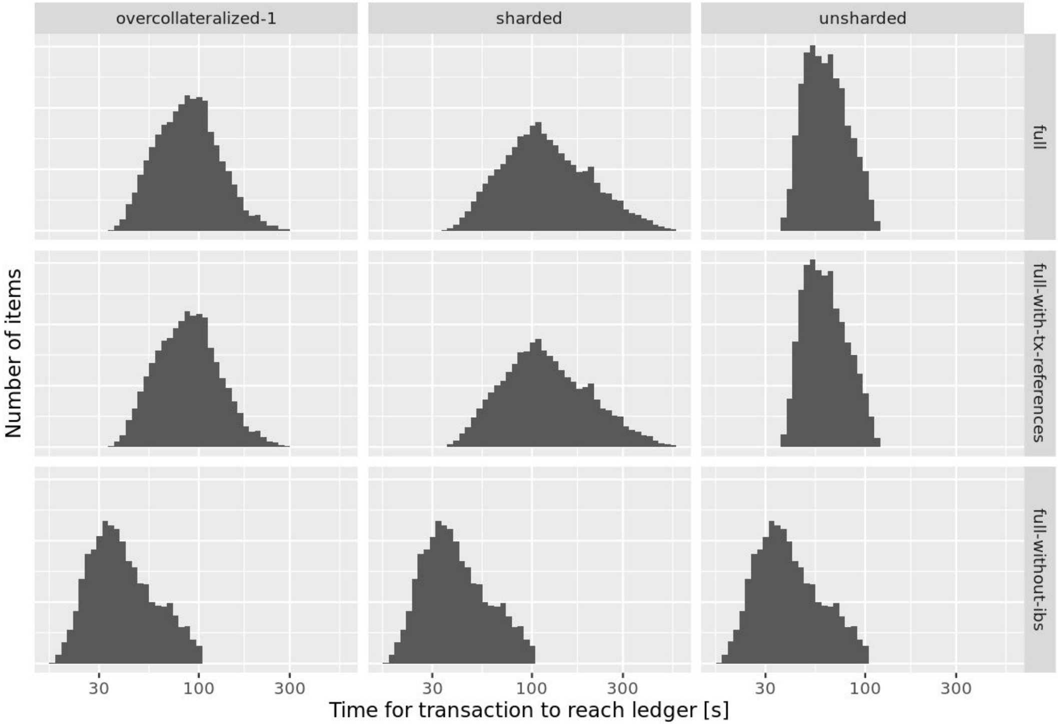
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage



Temporal efficiency

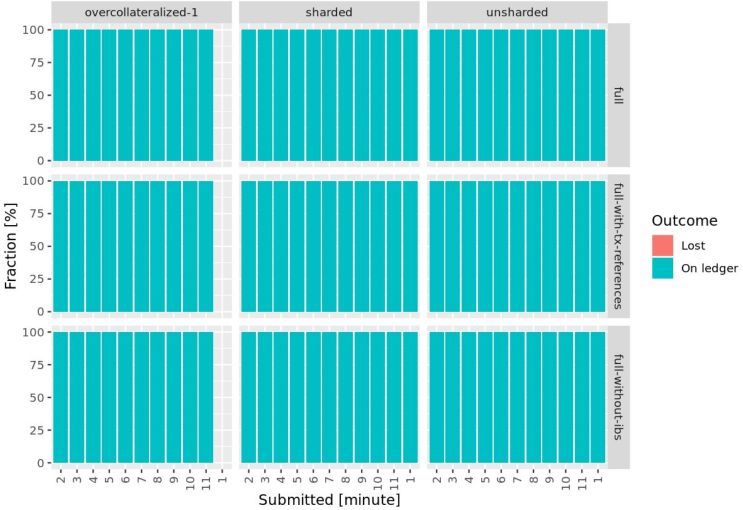
Time for transaction to reach the ledger

Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage

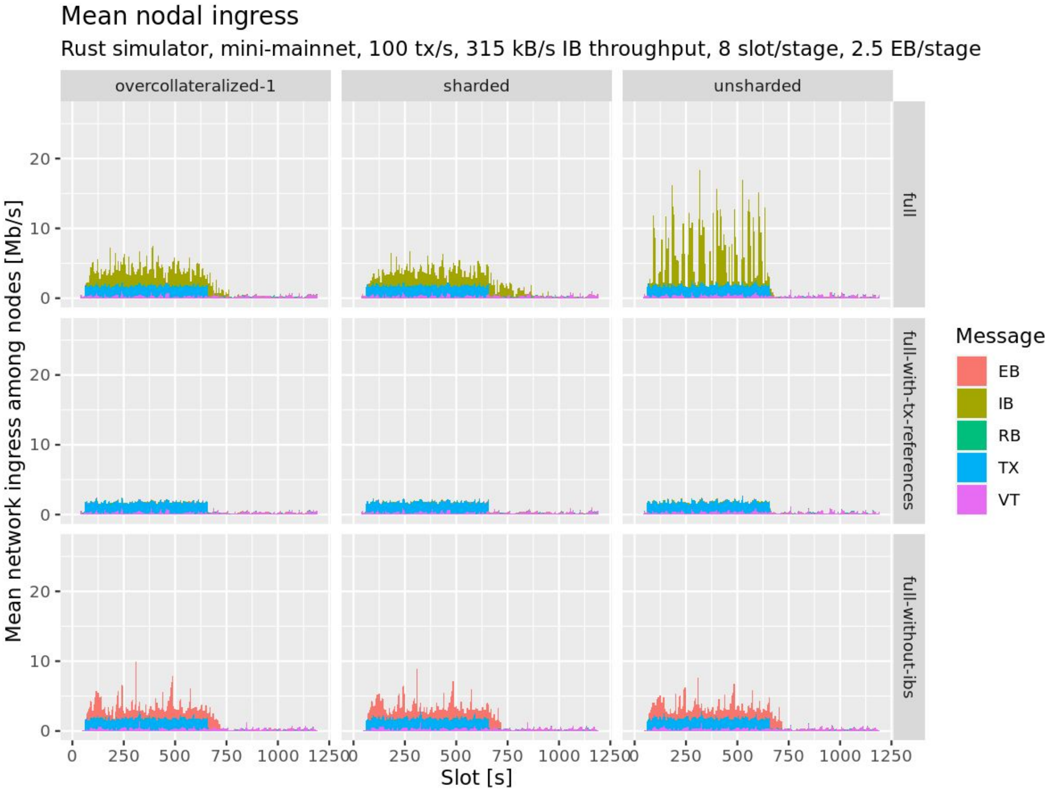


Transactions reaching the ledger

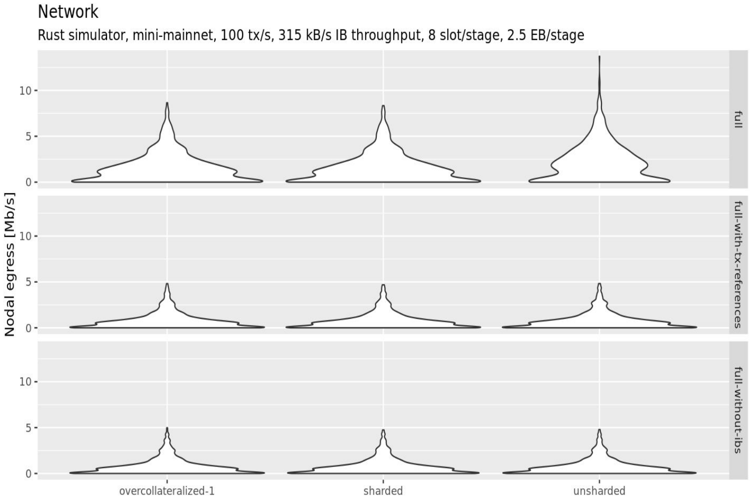
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage



Network

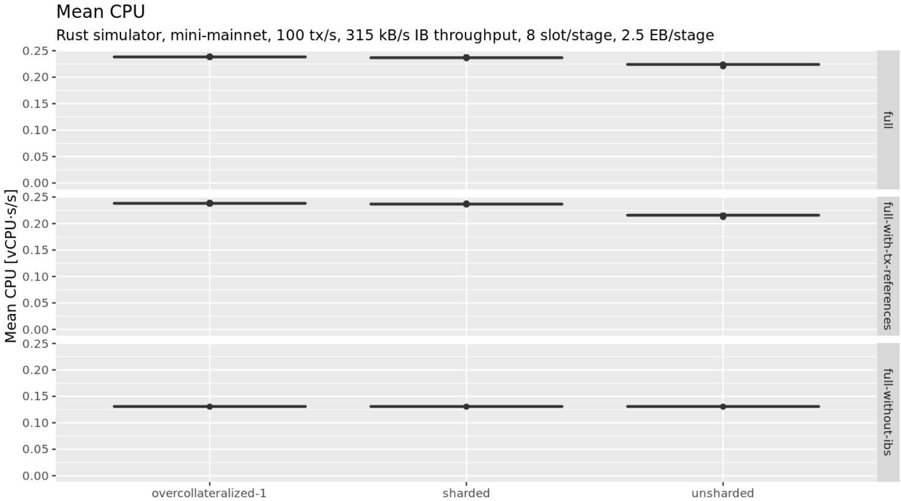


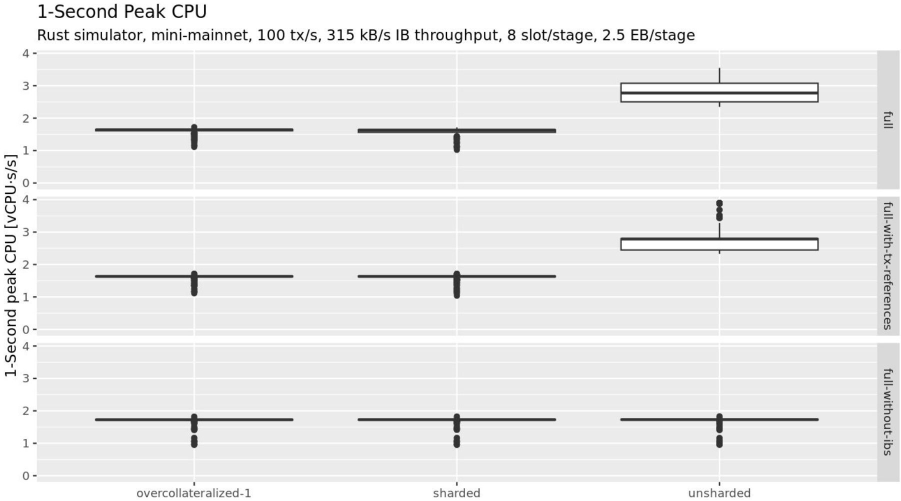
Mean nodal ingress
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage
overcollateralized-1
sharded
unsharded
full
full-with-tx-references
full-without-ibs
Mean network ingress among nodes [Mb/s]
0
10
20
0
250
500
750
1000
1250
Slot [s]
Message
EB
IB
RB
TX
VT



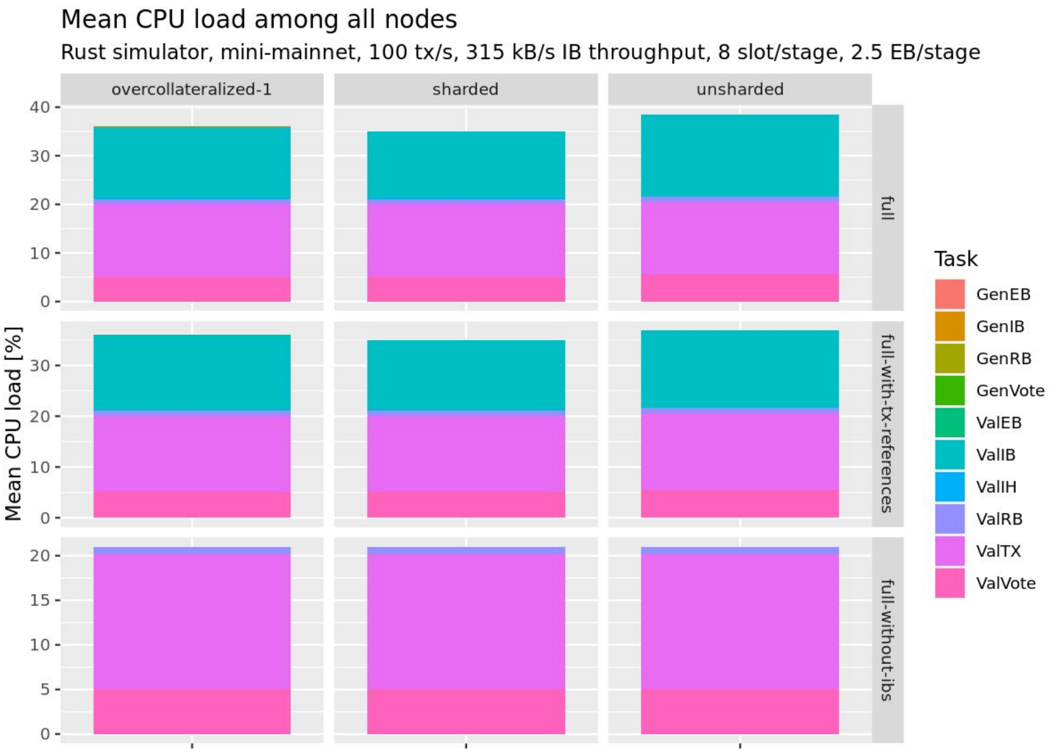
Network
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage
Nodal egress [Mb/s]
0
5
10
full
full-with-tx-references
full-without-ibs
overcollateralized-1
sharded
unsharded

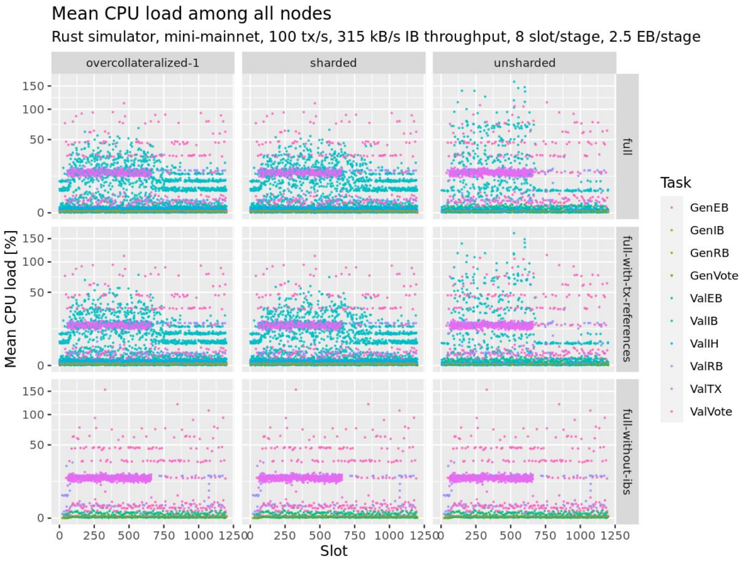
CPU


Mean CPU
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage
Mean CPU [vCPU·s/s]
0.00
0.05
0.10
0.15
0.20
0.25
full
full-with-tx-references
full-without-ibs
overcollateralized-1
sharded
unsharded


1-Second Peak CPU
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage
1-Second peak CPU [vCPU·s/s]
0
1
2
3
4
full
full-with-tx-references
full-without-ibs
overcollateralized-1
sharded
unsharded

CPU details


Mean CPU load among all nodes
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage
overcollateralized-1
sharded
unsharded
full
full-with-tx-references
full-without-ibs
Mean CPU load [%]
0
10
20
30
40
5
15
Task
GenEB
GenIB
GenRB
GenVote
ValEB
ValIB
ValIH
ValRB
ValTX
ValVote

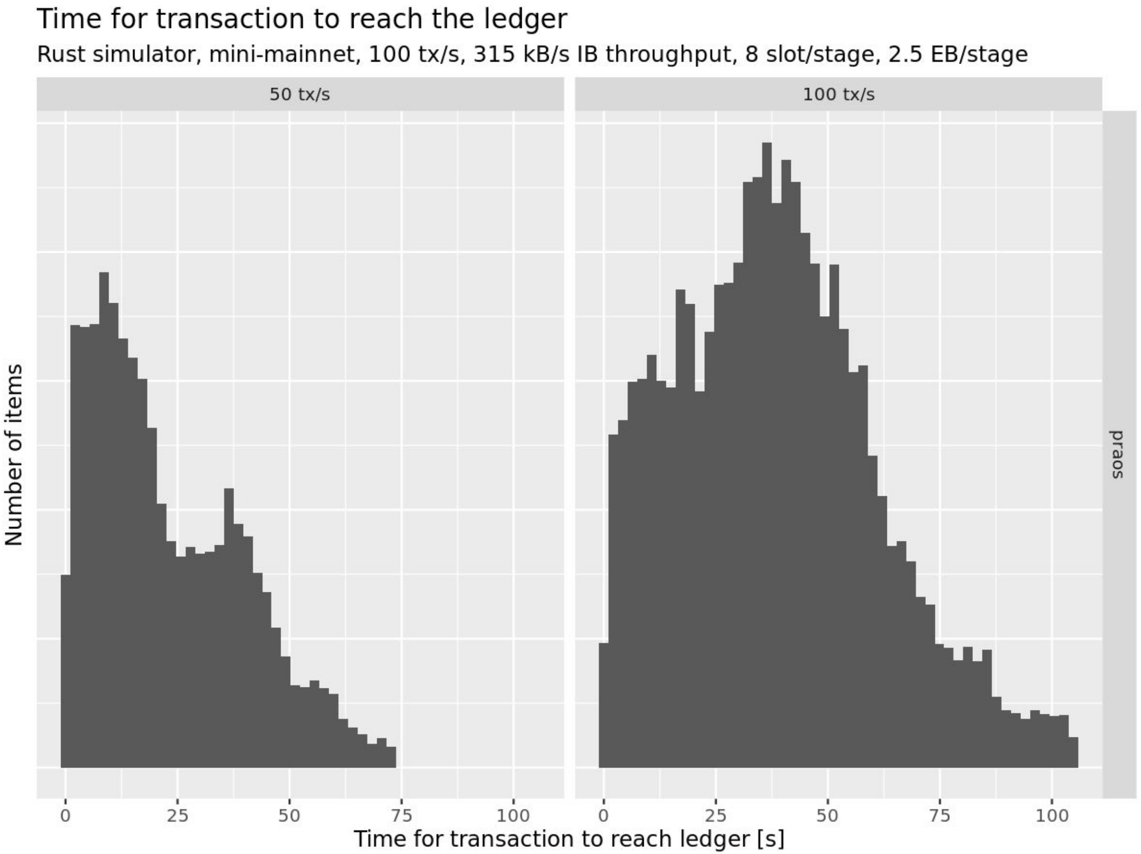

Mean CPU load among all nodes
Rust simulator, mini-mainnet, 100 tx/s, 315 kB/s IB throughput, 8 slot/stage, 2.5 EB/stage
overcollateralized-1
sharded
unsharded
full
full-with-tx-references
full-without-ibs
Mean CPU load [%]
0
50
100
150
Slot
0
250
500
750
1000
1250
Task
GenEB
GenIB
GenRB
GenVote
ValEB
ValIB
ValIH
ValRB
ValTX
ValVote

Praos with large blocks

Praos with large blocks

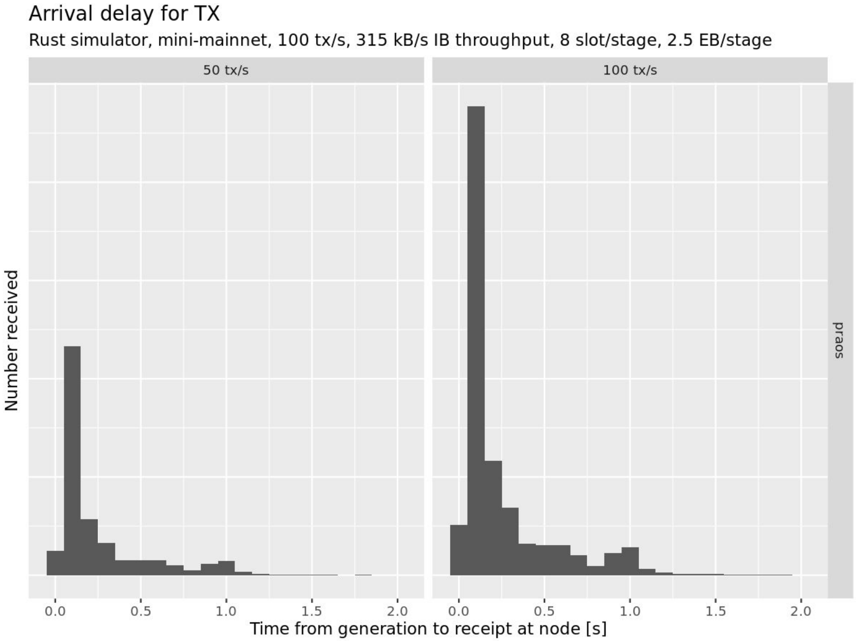
- What happens when Praos is run at 50 or 100 TPS?
- Maximum block size should be approximately
  - 3,145,728 bytes at 50 TPS
  - 6,291,456 bytes at 100 TPS
- How does Praos perform compared to Leios at these throughputs?

Transactions

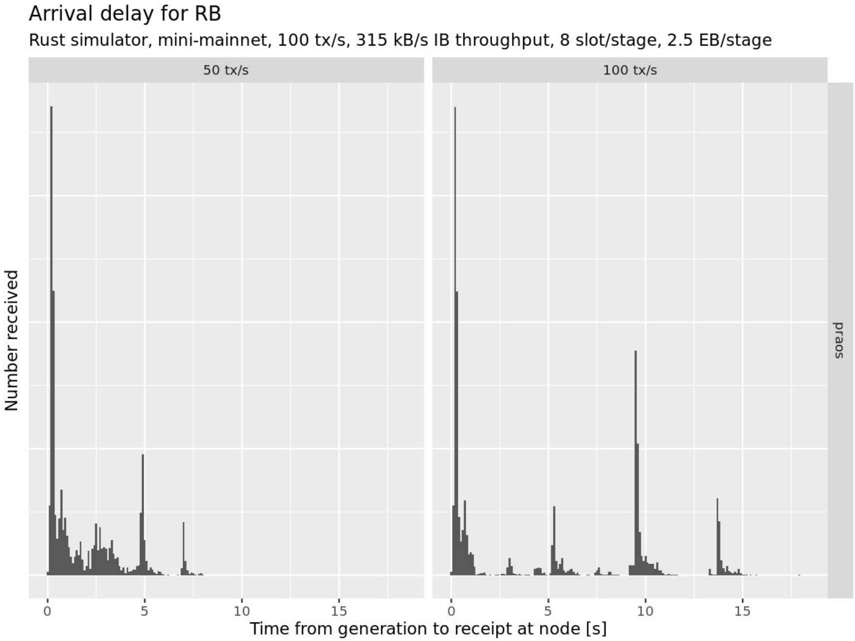


Transactions reach the ledger more quickly than for corresponding Leios scenarios.

Diffusion

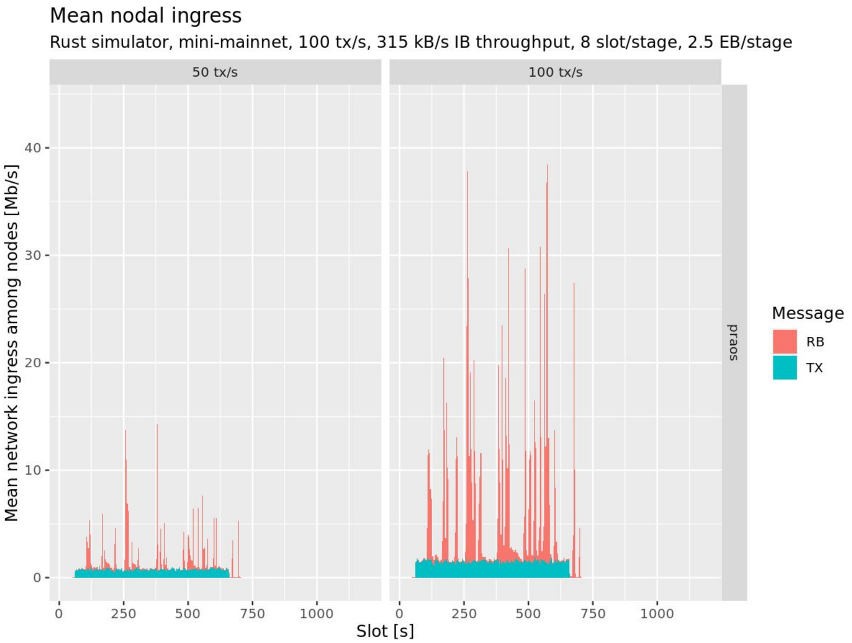


Transactions diffuse rapidly.

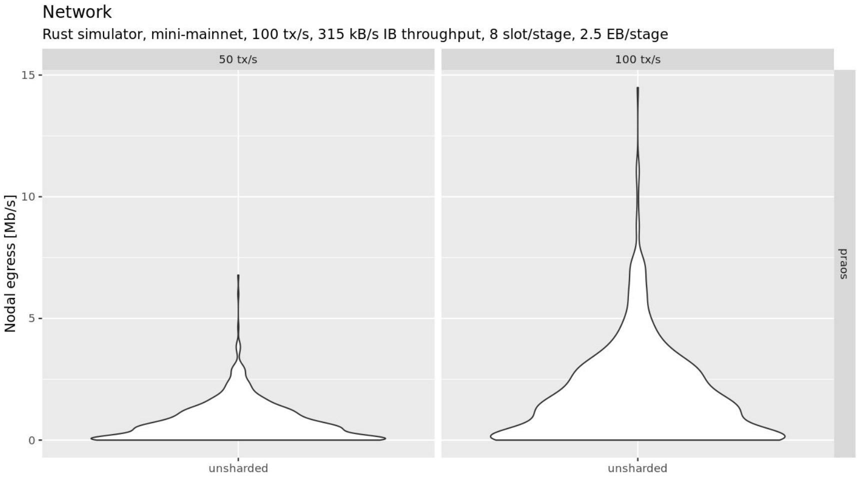


Blocks somes are greatly delayed.

Network

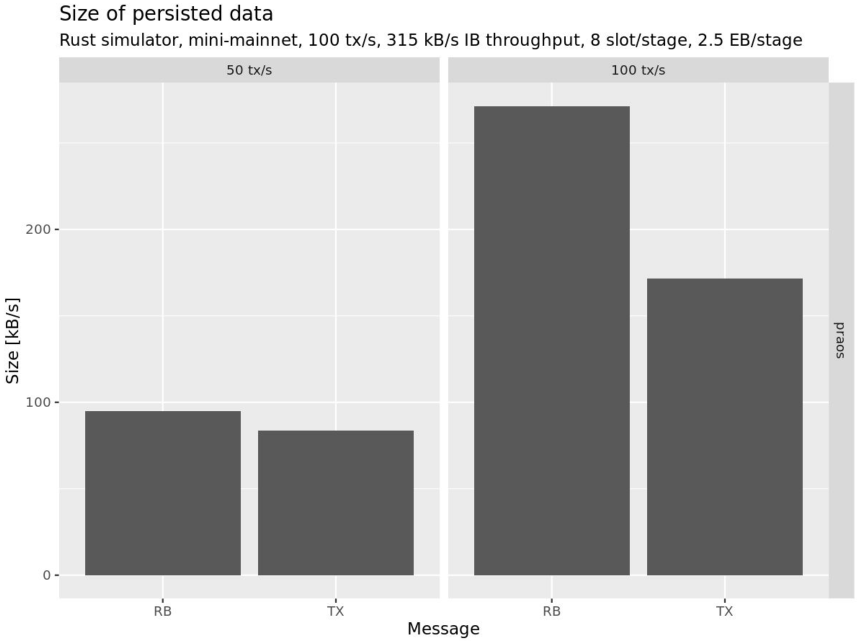


Block diffusion creates strong bursts of network traffic.

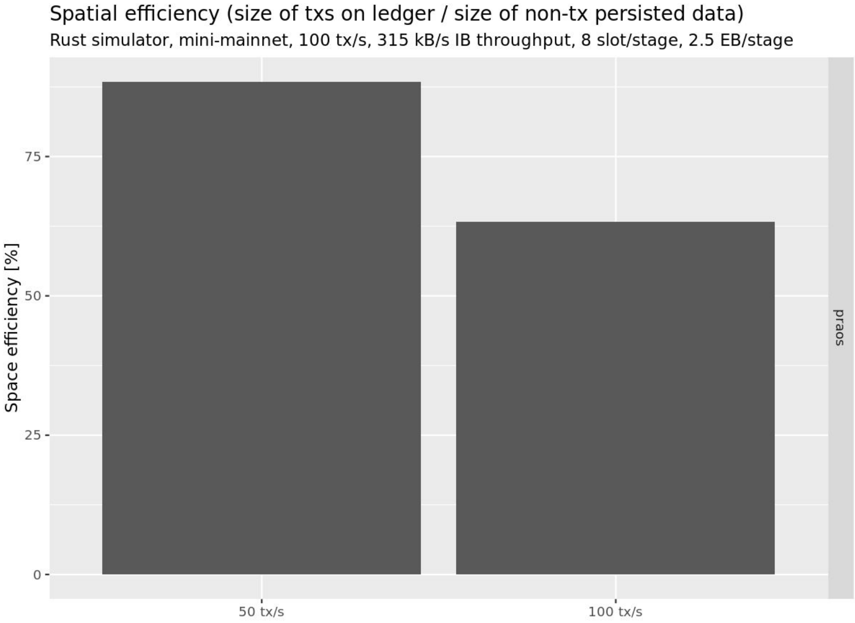


Bandwidth usage is less than inter-datacenter limits.

Data size

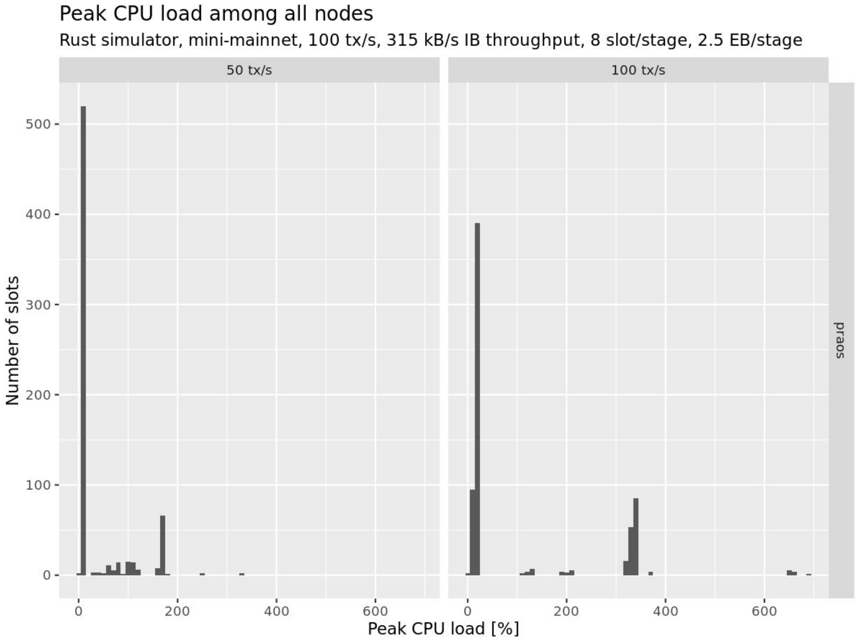


Forks waste space at high throughput.

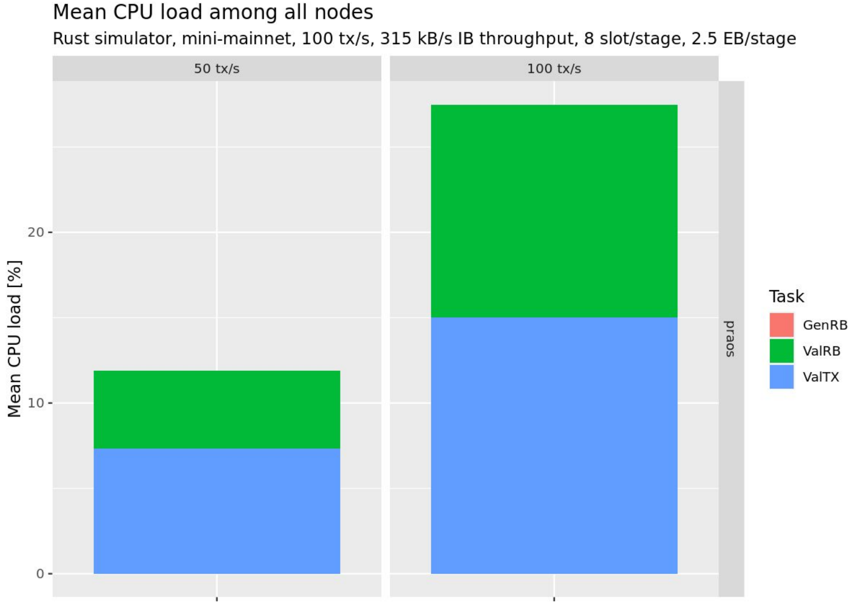


Spatial efficiency is only competitive with Leios at lower throughput.

CPU



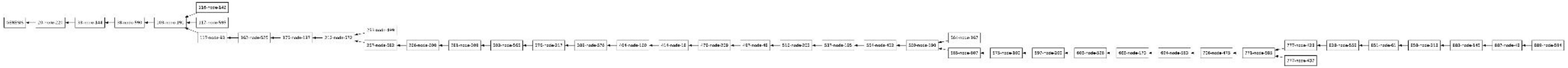
Block production and validation creates periodic bursts of CPU usage.



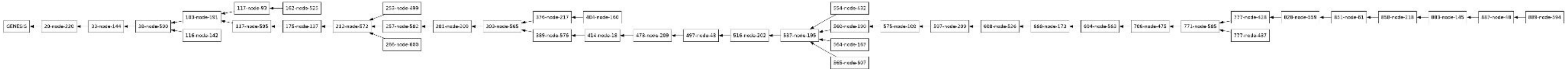
Mean CPU usage is less than corresponding Leios scenarios.

Forks

50 TPS

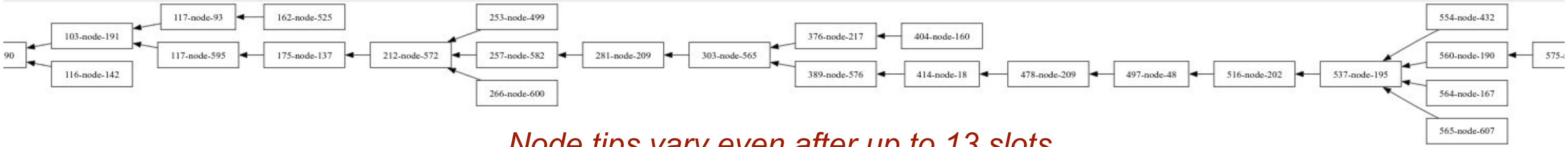


100 TPS



- *At 50 TPS, Praos operates normally, without congestion-induced forks.*
- *At 100 TPS, slow diffusion of blocks induces longer forks.*

100 TPS



Node tips vary even after up to 13 slots.

Findings

Praos vs Leios

- Leios simulations
  - Full-unsharded approach has an unacceptable spatial efficiency BUT its full-with-tx-references looks very promising.
  - Any sharded approach has the best spatial efficiency but worst latency (time to ledger/ RB).
  - Any over-collateralized approach is as good as sharded approach with an arguably unnoticeable spatial efficiency difference but a noticeable better/ shorter time to RB.
  - The transaction reference variant as expected further improves spatial efficiency with largest improvement in the unsharded variant.
  - Attack surface differs among variants, but not yet quantified.
- Praos simulations
  - Perform well at 50 TPS.
  - Exhibit congestion-induced forking at 100 TPS.
  - All transaction reach the ledger.
  - Bursty usage of network and CPU does not tax infrastructure resources.
  - Should we request cardano-node benchmark studies of large Praos blocks?