XI'AN · HYPERSCALE FOR RADIX

Hyperscale Weekly

Split the work, not the world

WEEK #19
14–20 Sept 2026

CURRENT MILESTONE

Milestone 2 · Radix Engine

Porting the Radix Engine so Scrypto runs across shards

Week 7 · since August 2026

Previously: flightofthefox had rewritten the VM execution model around a single core boundary — the largest architectural change of Milestone 2 so far.

Building on last week's VM rewrite, flightofthefox shifted into a massive pricing and resource-budgeting push — the fee system that lets the network charge transactions for exactly what they cost. The week also saw the account recovery model, block sync, and cross-shard settlement all take major steps forward. A heavy, broad week.

408	+18k −9.1k	600	20
+48% vs last wk	-21% vs last wk		
COMMITS	LINES CHANGED	FILES TOUCHED	DAY COMMIT STREAK

COMMITS / DAY 

peak 120/day · 6/7 days active

WHAT FLIGHTOFTHEFOX BUILT

Transactions priced by what they touch

flightofthefox wired per-leaf read costs, per-event framing charges, and scan pricing into execution — and moved the price table onto the beacon. Transactions now pay for exactly the resources they consume.

'see "The Governor" on hyperscale.rs

One intent to compose them all

The authorization model was unified around a single recursive intent — every transaction is one intent that composes others, each signed and nullified independently, eliminating scattered routing types.

'see "The Journey" on hyperscale.rs

Account recovery roles defined

New VM roles — a primary, a veto role, and a second factor — let an account freeze, amend guardians, or instantly rotate, all governed by on-chain rules rather than trust.

'see "The System" on hyperscale.rs

THE BOTTOM LINE

STANDOUT CHANGE

The fee and resource-pricing system was wired end-to-end — from per-leaf read costs to the beacon's price table — the biggest push toward making transactions pay their own way.

WHAT'S NEXT

Likely next: continuing to wire the fee system into the execution lifecycle and pushing the intent model toward feature parity with Babylon's component model.

More of what flightofthefox built

The week had more stories than one page could hold.

THE STORIES, CONTINUED

Sync frontiers and halted tips

Block sync moved into a proper state machine, and halted-tip recovery was rebuilt to harvest from verified timeouts and screen committee membership before committing.

'see "The Archive" on hyperscale.rs

Cross-shard settlement tightens

Provision windows are now shared rather than mirrored, dropped bundles satisfy their askers, and unjudged fees settle against what the payer holds — tightening the commit path.

'see "The Generals" on hyperscale.rs

Departed shards retired cleanly

Shard termination rules hardened: departed shards retire when their evidence window closes, terminated chains stop composing ticks, and expired pools are bounded.

'see "The Will" on hyperscale.rs

ALSO THIS WEEK

- Wallets can now ask a node for a read-only preview of what a transaction would do before submitting it.
- Shard splits can now trigger automatically when a shard runs near its capacity limits.
- Network delay is estimated from the committed chain, driving round timers and fetch timeouts.
- Quorum timestamps are now taken as the median of voter clock readings, smoothing the attested clock.
- A simulated Byzantine sender can now rewrite notifications and gossip, strengthening fault injection.
- Retention costs are now weighed by the actual bytes a validator keeps.
- Each shard attests its own emission share and is verified on every shard.
- Messages must now state their class explicitly rather than defaulting, improving network prioritization.

From the community

What people are saying, and the words behind the work.

HEARD IN THE CHAT

The mood: Heated community debate about Hyperscale's future relationship to Radix, including a long proposal from Wim about a possible funded restart. flightofthefox stayed focused on honest trade-offs.

> *"blockchains can't be the best across every dimension simultaneously. you have to pick and choose your battles"*
>
> — FLIGHTOFTHEFOX, IN THE COMMUNITY TELEGRAM

FLIGHTOFTHEFOX EXPLAINED

Hyperscale's core trade-off

Finality is the trade-off — it's always quicker to execute in blocks than go through async execution for cross-shard. If you don't need more than one shard, it's a strict downgrade.

Who should use Hyperscale

Literally zero point using it unless you have a plausible situation with hundreds of thousands or millions of state transitions.

Long-term plans

I'll work on other things after it's stable and just maintain it. Can't work on open source forever unless networks sponsor it.

Market positioning

The market for transactions that can tolerate a few extra seconds is larger than the market for sub-second finality — until low-latency chains run out of blockspace, the benefits of shards will be non-obvious.

THIS WEEK'S CONCEPT

The Governor — *A governor, not a vote.* The dominant work was pricing: per-leaf reads, per-event framing, scan costs, and moving the price table onto the beacon — the economic layer that makes every transaction pay its own way.

hyperscale.rs/governor

JARGON, DECODED

Intent — A signed authorization that names which accounts and state a transaction touches, now unified into one recursive structure.

Leaf — A single data entry in the state tree; the unit the fee system charges for reading or writing.

Preview — A read-only dry run that tells a wallet what a transaction would do before it is submitted.

Provision — A piece of state one shard sends another, proven against a committed block.

ZOOM OUT

The bigger picture

Where the work landed, and the map to read along on hyperscale.rs.

MOMENTUM · LINES CHANGED / WEEK

last 8 weeks · newest right

THIS WEEK'S WORK

86% Rust 14% Config

WHERE THE WORK LANDED

node · The Archive
sync FSM, halted-tip recovery, provisions

vm/effects · The Governor
fee pricing and resource budgeting wired in

vm/sdk · The Journey
intent model and account authorization

vm/harness · The Crash Lab
fuzz tests and CI blob alignment

shard · The Will
shard termination and departed retirement

THE FULL PICTURE

The System
The whole sharded design at a glance.

The Journey
One transaction's trip through the network.

The Clock
Consensus-attested time across independent shards.

The Census
The leaderless beacon of validators, stake and shards.

The Overlap
Why two conflicting blocks can never both commit.

The Generals
All-or-nothing cross-shard commits, computed not voted.

The Archive
Every published byte is preserved or provably expired.

The Library
All state in one merkle tree; a shard is a subtree.

The Will
How in-flight transactions settle when a shard dies.

The Lottery
Random, ever-moving committees; proven cheats are jailed.

The Triage
Under overload, urgent traffic never waits for bulk.

The Governor
Validator entry is re-priced every epoch, like a market.

The Crash Lab
The simulator that replays any failure byte-for-byte.

The Proof
Proving safety with maths, not just tests.

The Asterisks
Every design's trade-offs — including Hyperscale's own.

THE ROAD TO MAINNET

M1	M2	M3	M4	M5	M6
Adaptive Sharding	Radix Engine	Gateway & API	Validator GUI	Migration	Live support
done	~5 mo	~3 mo	~3 mo	~3 mo	12 mo

Now: Milestone 2 · Radix Engine — Week 7 · since August 2026. Since day one: 4,918 commits — one developer, fully in public.

LINKS

WEBSITE
hyperscale.rs

DISCUSS THIS ISSUE ON X
x.com/i/status/2101899653344079942

CODE (PUBLIC)
github.com/hyperscalers/hyperscale-rs

MILESTONE PROPOSAL (PDF)
hyperscale.rs/xian_proposal.pdf

COMMUNITY (TELEGRAM)
t.me/hyperscale_rs

AUTHOR
flightofthefox

scan for hyperscale.rs

Questions? flightofthefox is always happy to discuss in the community Telegram — t.me/hyperscale_rs.

Written by AI from public commits, the community chat and hyperscale.rs — may contain mistakes.