XI'AN · HYPERSCALE FOR RADIX

Hyperscale Weekly

Split the work, not the world

WEEK #20
21–27 Sept 2026

CURRENT MILESTONE

Milestone 2 · Radix Engine

Porting the Radix Engine so Scrypto runs across shards

Week 8 · since August 2026

Previously: flightofthefox had wired the fee and resource-pricing system end-to-end and unified the intent model, pushing deeper into Milestone 2's VM work.

Week 8 of Milestone 2 brought a deep, focused rebuild of how cross-shard transactions are tracked and resolved inside the VM. flightofthefox reworked the entire 'crossing' system — the machinery that lets a transaction spanning multiple shards record, answer, and clean up its commitments — while also tightening fee holds and pruning dead execution paths. In Telegram, a rich conversation about migration and compatibility showed the community engaging seriously with what Hyperscale would mean for existing Radix apps.

32	+3.1k −3.4k	128	28
-92% vs last wk	-76% vs last wk		
COMMITS	LINES CHANGED	FILES TOUCHED	DAY COMMIT STREAK

COMMITS / DAY  peak 10/day · 7/7 days active

WHAT FLIGHTOFTHEFOX BUILT

Cross-shard crossings rebuilt

Most of the week redesigned 'crossings' — how the VM tracks cross-shard transactions end to end. Crossings are now named once with all keys derived from that name, answers carry two verdicts, and blocks carry crossing records directly.

'see "The Generals" on hyperscale.rs

Fee holds and vault deposits

Fee reservations are now treated as committed holds inside the kernel, and vault deposits moved from scattered jobs into a single kernel path — making resource handling cleaner and more predictable.

'see "The Journey" on hyperscale.rs

Dead execution paths pruned

Several outdated job types and a crossing tombstone mechanism were removed entirely, replaced by simpler flag-based logic — less code to maintain, fewer edge cases to break.

'see "The System" on hyperscale.rs

THE BOTTOM LINE

STANDOUT CHANGE

The crossing system — how cross-shard transactions are named, tracked, answered, and cleaned up — was comprehensively rebuilt.

WHAT'S NEXT

Likely next: continuing to refine the crossing system and tackling the local execution path flightofthefox flagged as a major challenge.

From the community

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

HEARD IN THE CHAT

The mood: A lively week: a developer named Sebööõl dove into the codebase and sparked a detailed exchange about migration, contract compatibility, and whether Babylon and Hyperscale can coexist.

> *"leg local execution is my vietnam"*
>
> — FLIGHTOFTHEFOX, IN THE COMMUNITY TELEGRAM

FLIGHTOFTHEFOX EXPLAINED

Migration path to Hyperscale

The rough plan is dump, transform, genesis — contracts rebuilt before launch if devs are active, stragglers upgraded later through beacon-chain voting with blueprints sitting disabled until then.

Component interaction design

Calls between components should work if the target is in immutable config or passed as an argument, but he wants to explore extending the manifest DAG at the call site rather than replicating Radix Engine's method dispatch.

Contract upgrades without proxies

He plans a native blueprint upgrade system so developers don't have to build proxy patterns, with upgrades coordinated through beacon-chain witnesses to apply consistently at the same epoch.

Why no public backlog

The foundation is still in flux, so maintaining a formal spec would slow things down — there is an order of operations, and it's too early to size the migration.

THIS WEEK'S CONCEPT

The Generals — *The two generals put down their messengers.* The dominant work this week was rebuilding how cross-shard transactions are tracked and resolved — the core of atomic cross-shard commitment.

hyperscale.rs/generals

JARGON, DECODED

Crossing — The VM's name for a cross-shard transaction's tracked lifecycle as state moves between shards.

Fee hold — The portion of a transaction's fee reserved upfront before it runs, so it cannot overspend.

Tombstone — A marker showing a cross-shard record has been retired and is safe to clean up later.

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

88% Rust · 7% Config · 5% Docs

WHERE THE WORK LANDED

vm/effects · The Generals
crossing answers and state effects reworked

vm/kernel · The Generals
vault deposits, fee holds, crossing jobs

vm/sdk · The Journey
APIs updated for new crossing and fee model

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 8 · since August 2026. Since day one: 4,959 commits — one developer, fully in public.

LINKS

WEBSITE
hyperscale.rs

DISCUSS THIS ISSUE ON X
x.com/i/status/2104436407494185108

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.