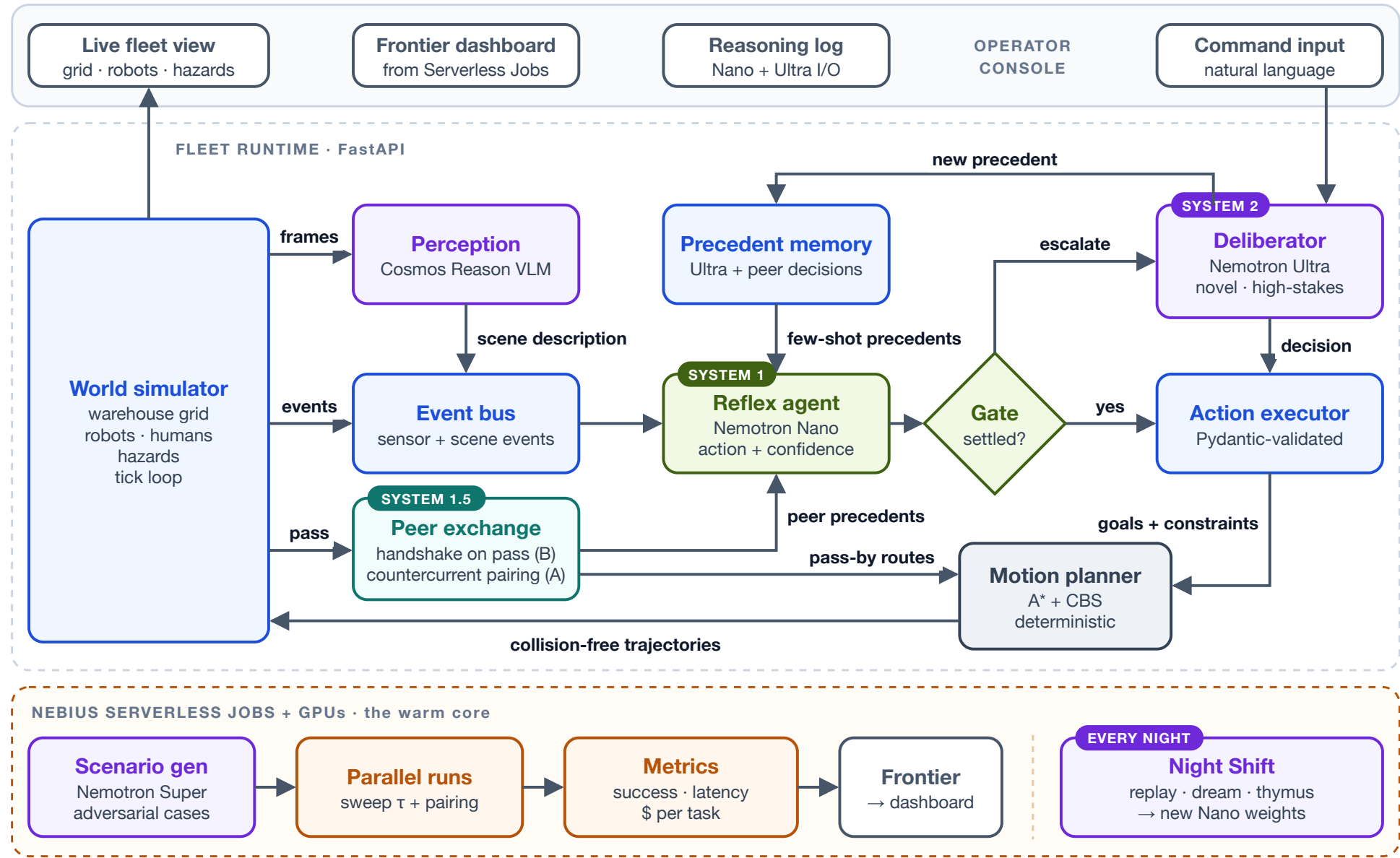


NemoFleet · System 1 / System 2 for robot fleets

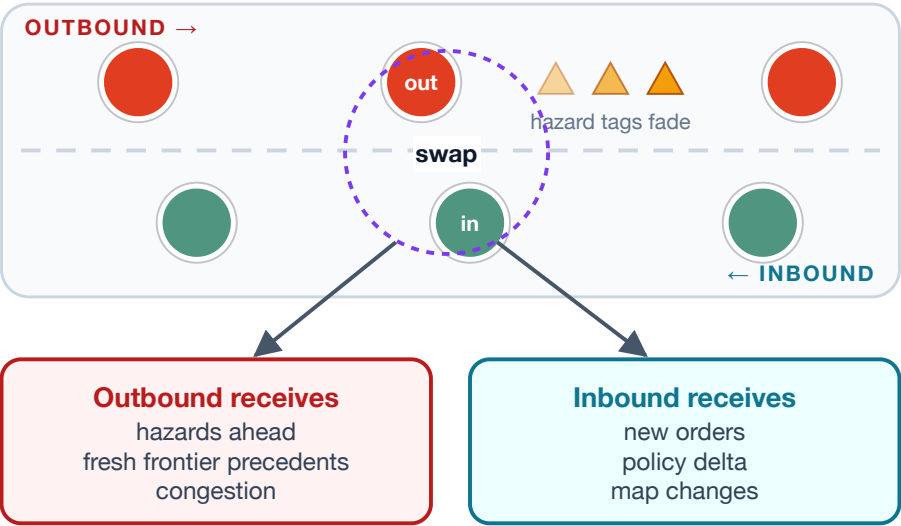
A fast model on each robot handles routine events, peers share what the fleet already knows when they pass, and a large reasoning model in the cloud handles only what is novel or high-stakes. A deterministic planner owns every trajectory.



Robots teach each other in passing

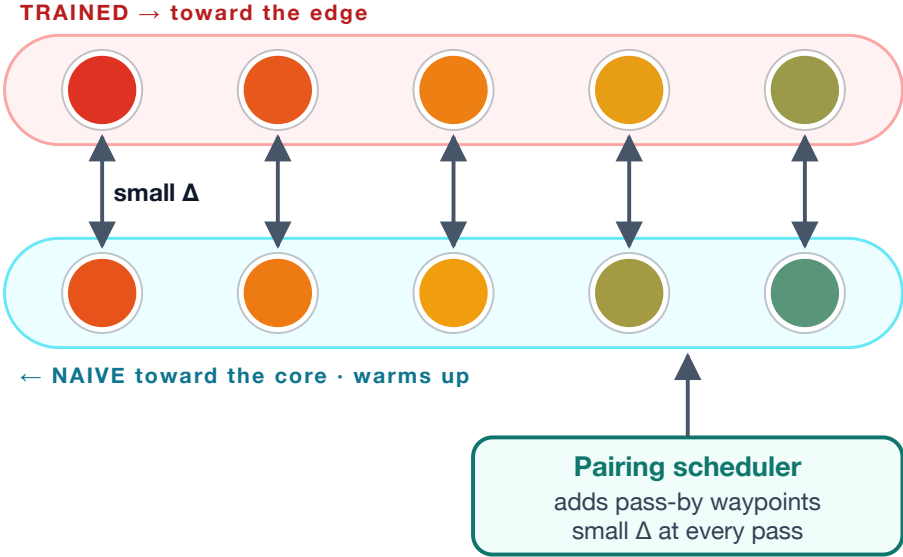
Two designs from the countercurrent options work together. **B** moves knowledge at runtime: robots swap a small payload whenever they pass. **A** shapes who passes whom, so each robot meets peers only slightly ahead of it. Most already-seen problems get settled without a cloud call.

B · Handshake on pass (runtime)



No model call · bounded payload · Cosmos called once per hazard

A · Countercurrent pairing (learning)



Peers never override safety

A human present, an unknown hazard class or a blocked exit always escalates to Ultra.

Provenance on every precedent

Who decided it (Ultra or a peer), when, and how many hops. Confidence decays per hop.

Everything expires

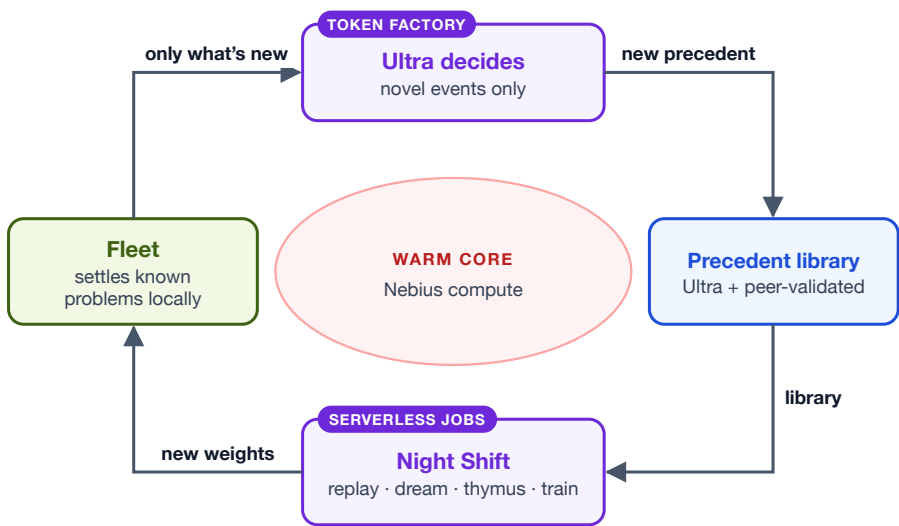
Precedents and hazard tags carry a TTL, so a bad lesson can't circulate forever.

Conflicts escalate

Ultra outranks peers. Two precedents that disagree are themselves an escalation trigger.

We don't cut compute. We move it.

A duck keeps heat in its core and lets its feet run cheap. NemoFleet does the same: edge robots settle known problems cheaply, and Nebius compute goes to what only the core can do: deliberating on new problems, stress-testing the fleet at scale, and learning from experience.



WORKLOAD	RUNS ON	GROWS WITH
Deliberation on novel events	Token Factory · Ultra	new sites, new hazards, fleet size
Reflex inference	Token Factory · Nano	robots × events
Perception, first sighting	Token Factory · Cosmos	new hazards
Red-team + pairing sweeps	Serverless Jobs	scenarios × τ × strategies
Night Shift	Batch + Serverless Jobs	incidents × dreams × fleets

Knowledge moves into weights

Precedents start in the context window; distillation fine-tunes them into Nano, so every new robot starts warm.

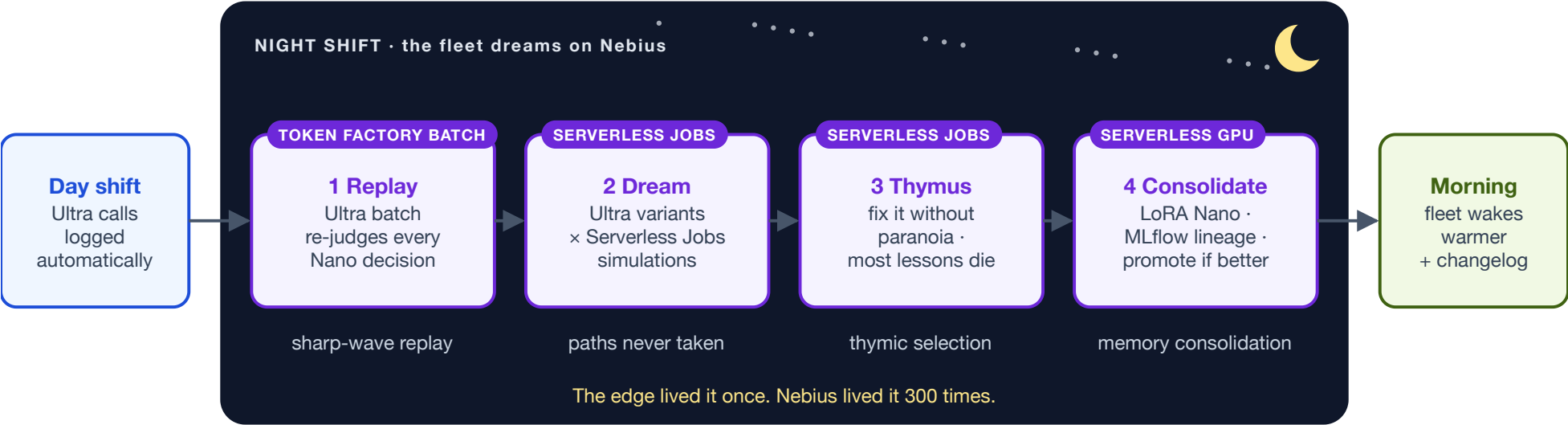
Gate before rollout

A distilled Nano ships only if it beats the current one on the red-team suite, run on Serverless Jobs.

Bigger fleets, same budget

Cheaper decisions per robot let operators deploy more robots, so total usage grows with the business.

Animals consolidate memory in sleep: the hippocampus replays the day’s key moments, including paths never taken, and the thymus deletes cells that would attack the body. Every night NemoFleet does the same on Nebius. The edge experiences an incident once; the core lives through hundreds of versions of it.



Day 1 · the incident

A robot nearly hits a spill at a blind corner and escalates to Ultra. Every Ultra call is already logged.

Night · on screen

Live counters: decisions re-judged, dreams simulated, lessons through the thymus, fine-tune, promotion gate. Real runs only.

Day 2 · the payoff

A different robot meets a variant it never saw (spill plus forklift) and handles it with Nano alone: “learned in dream #217”.

Verify in week 1

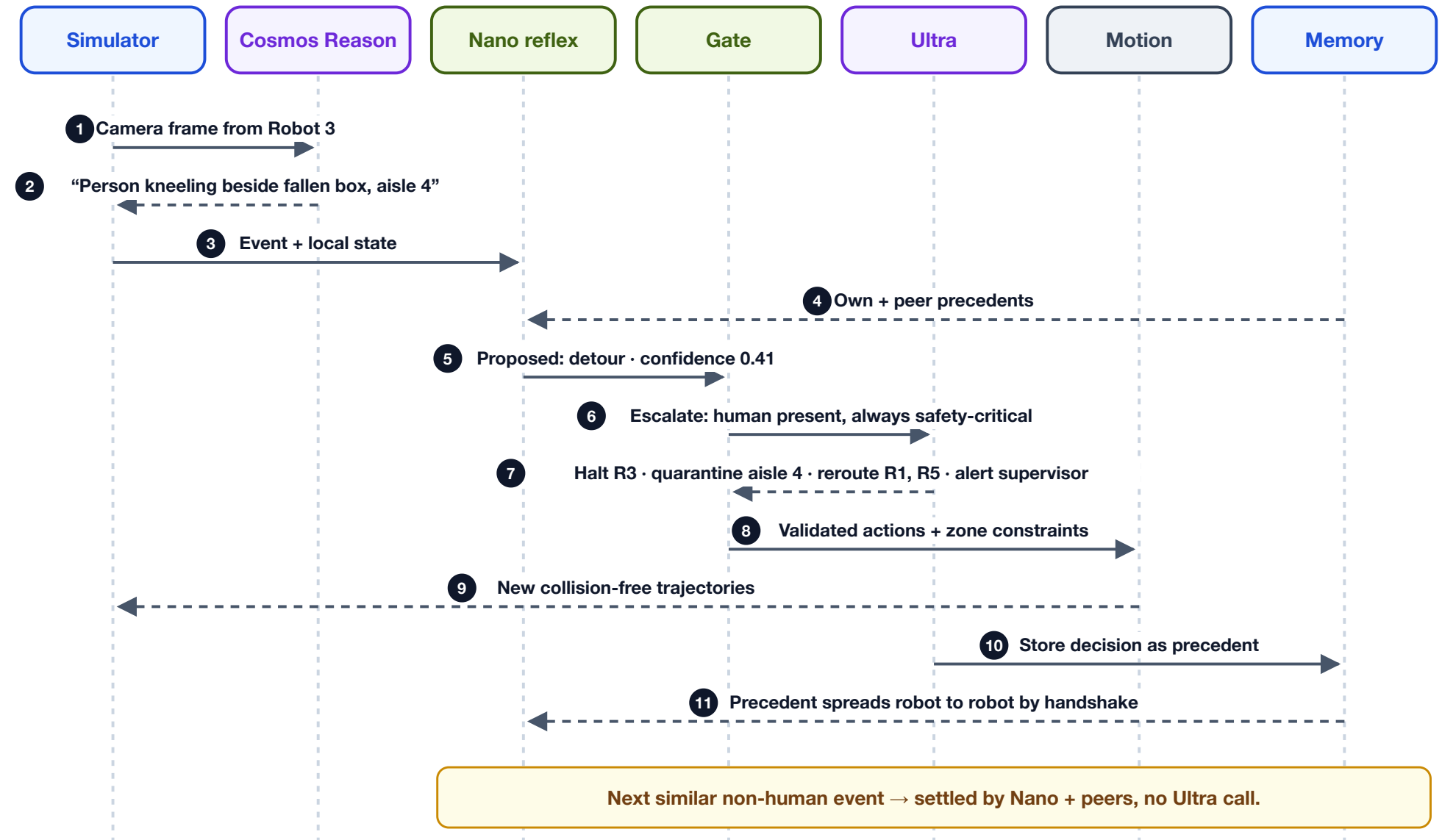
Nemotron models live in the catalog; batch inference for Ultra; Data Lab logging. Nemotron is not on Token Factory’s fine-tune list, so training runs in our own container on Serverless Jobs.

Stretch: physical trigger

A real Jetson Orin Nano spots a labeled spill and stops locally; Ultra + Tavily pull the Safety Data Sheet. That incident becomes tonight’s dream.

A person enters a robot lane

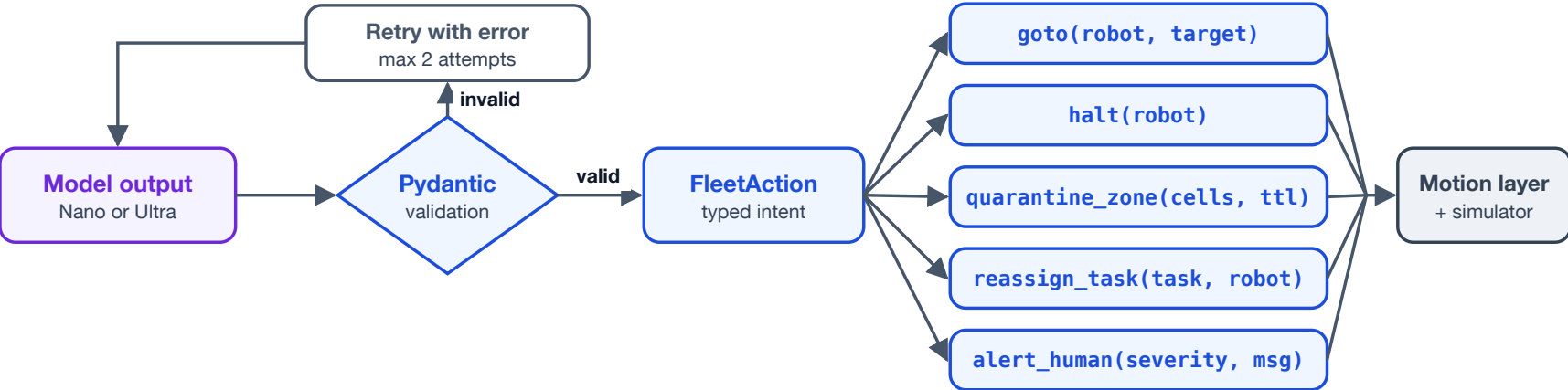
Humans are always safety-critical, so this escalates to Ultra; the decision then spreads robot to robot by handshake.



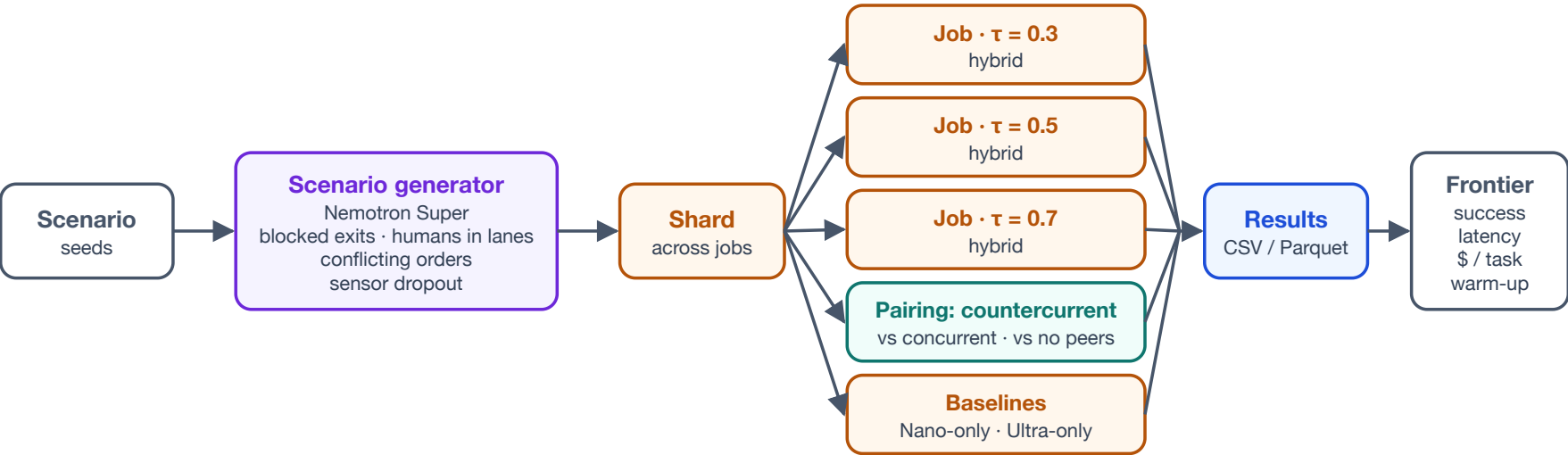
Principle: models decide *what* should happen; the planner decides *how* to move. Every model output must pass schema validation before anything moves.

LAYER	MODEL / ALGORITHM	DECIDES	LATENCY	NEVER DOES
Perception	Cosmos Reason	What the camera sees, on first sighting	~1 s	Plan actions
Reflex	Nemotron Nano	Routine responses + confidence	< 1 s	Handle novel safety events alone
Peer exchange	Handshakes, tags, pairing	What the fleet already knows, shared	< 50 ms	Override a safety escalation
Escalation gate	Rules + threshold τ	Settled locally, or System 2 needed	~0	Generate content
Deliberation	Nemotron Ultra	Novel, high-stakes decisions	5–30 s	Produce raw coordinates
Night Shift	Ultra batch + Jobs + LoRA Nano	What the fleet learns overnight	Offline	Promote a model that fails
Motion	A* + CBS	Collision-free paths + pass-by waypoints	< 10 ms	Interpret semantics
Executor	Pydantic schemas	Whether an action is valid	~0	Accept malformed LLM output

Action schema boundary



Serverless Jobs run adversarial scenarios in parallel across escalation thresholds τ and pairing strategies: the escalation frontier, plus countercurrent vs. concurrent vs. no-peer warm-up and hazard-spread curves.



Where NVIDIA and Nebius are used

COMPONENT	NVIDIA MODEL	NEBIUS SERVICE
Reflex agent	Nemotron Nano	Token Factory
Deliberator	Nemotron Ultra	Token Factory
Perception	Cosmos Reason	Token Factory or AI Cloud endpoint
Scenario generator	Nemotron Super	Token Factory
Peer exchange	None: no model call is the point	—
Night Shift: replay + dream	Nemotron Ultra	Token Factory batch inference · Data Lab
Night Shift: train	Nemotron Nano (LoRA)	Serverless Jobs (GPU) · MLflow · Serverless Endpoints
Red-team + pairing sweeps	—	Serverless Jobs

Model IDs to be confirmed against the Token Factory catalog.

models.py

WorldState, **Robot**, **Event**
FleetAction, **Decision**, **Precedent**

sim/

world.py — grid, robots, humans, tick loop
proximity.py — passes, handshakes, fading tags
scenarios.py — loading + generation

motion/

astar.py — single-agent paths
cbs.py — multi-agent conflict search

cognition/

client.py — Token Factory client
reflex.py — Nano · System 1
peers.py — payloads, provenance, TTL (B)
pairing.py — countercurrent scheduler (A)
deliberator.py — Ultra · System 2
gate.py — escalation + safety overrides
perception.py — Cosmos Reason
memory.py — precedent store

nightshift/

replay.py — batch hindsight review, τ
dream.py — counterfactual variants + sims
thymus.py — positive / negative selection
consolidate.py — LoRA Nano, gate, promote

api/

server.py — /state /command /inject /tick /log

jobs/

redteam.py — τ sweep + pairing sweep
nightshift.py — Night Shift pipeline entry
frontier.py — aggregation + charts

ui/

Operator console: live view, command input, reasoning log, frontier dashboard



Fleet runtime



System 1 · reflex



System 1.5 · peers



NVIDIA models (Token Factory)



Deterministic planning



Serverless Jobs