KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

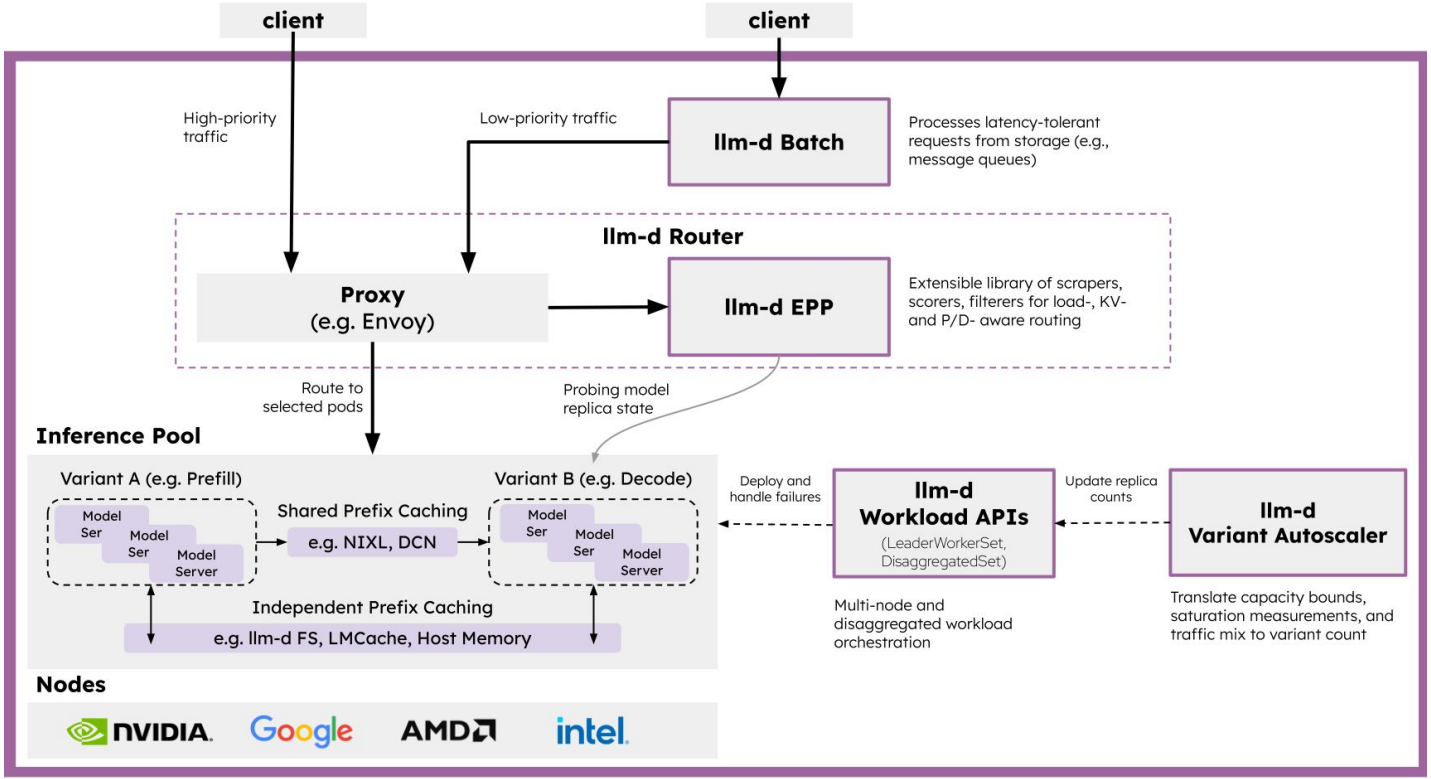
Simulate, Evolve, Validate, Ship: AI-Discovered Algorithms for LLM Serving Stacks

Jing Chen and Mert Toslali, IBM Research



What is llm-d?

SOTA Kubernetes-native distributed LLM inference



Advancing llm-d means inventing new strategies

Each subsystem is an open design problem with room for **better algorithm**

Routing

where does each request go?

Admission control

what enters under load?

Flow control

which admitted request next?

Scheduling + batching

how to configure KV-cache reuse?

Prefill/decode

where is each phase placed?

Autoscaling

when to grow or shrink?

Such complex platforms advance slowly

THE COST OF ONE EXPERIMENT

days

and thousands of dollars per accelerator sweep

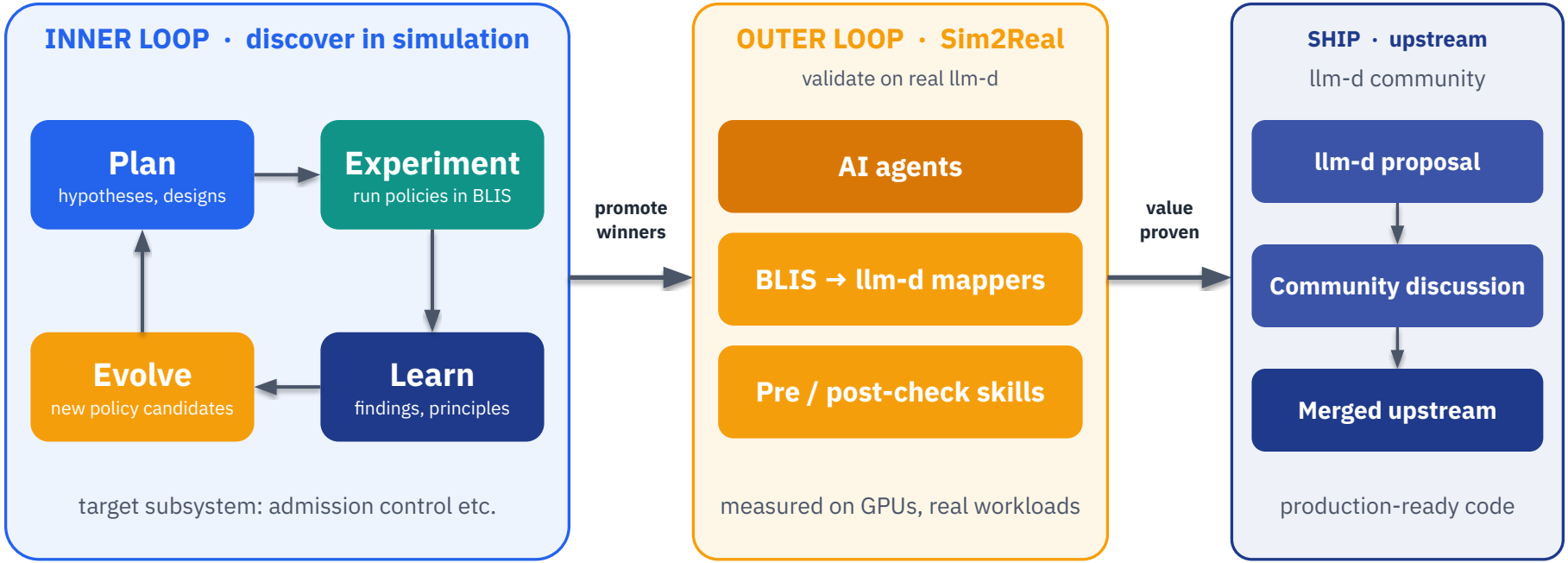
👉 **limited to a few ideas per week**

👉 **cannot test everything with speed**

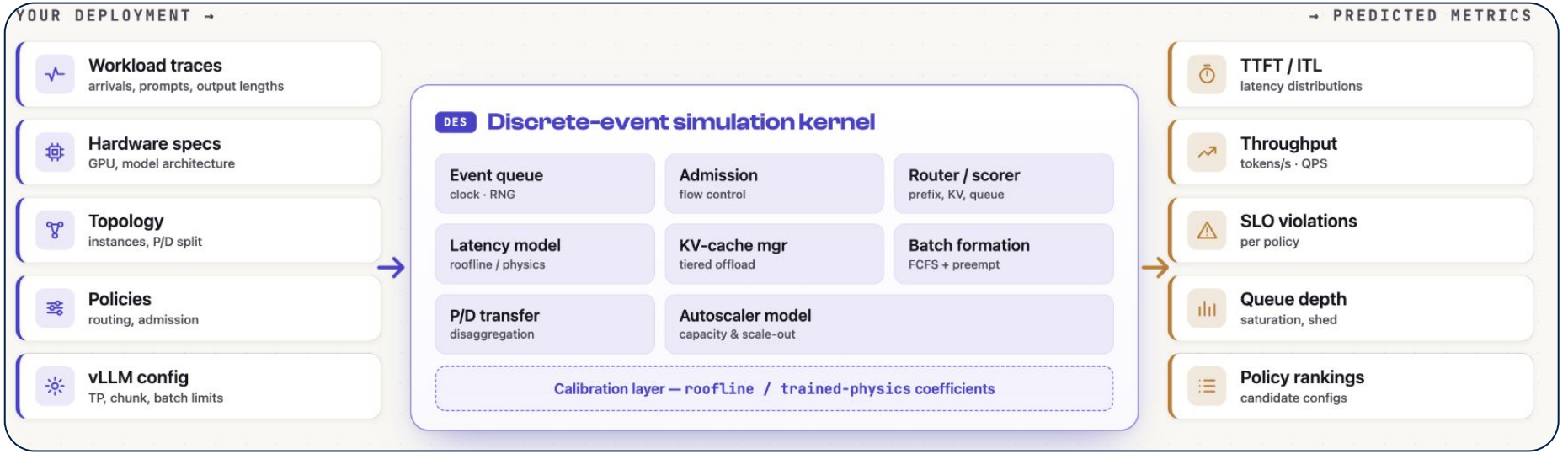
PROMISING STRATEGIES NEVER TRIED

- adaptive router
- proactive router
- priority flow control
- new P/D placement policy
- smarter autoscaler
- KV-aware batching

AI-powered evolution loop

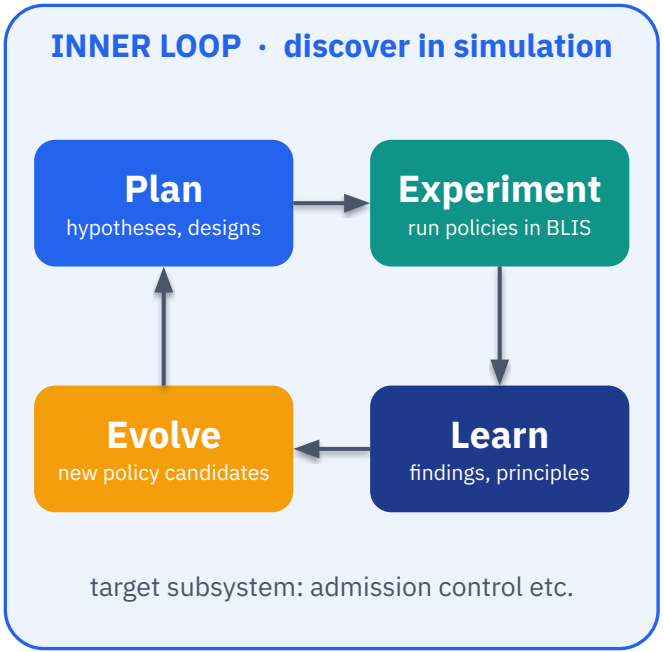


BLIS: full stack llm-d/vllm simulator

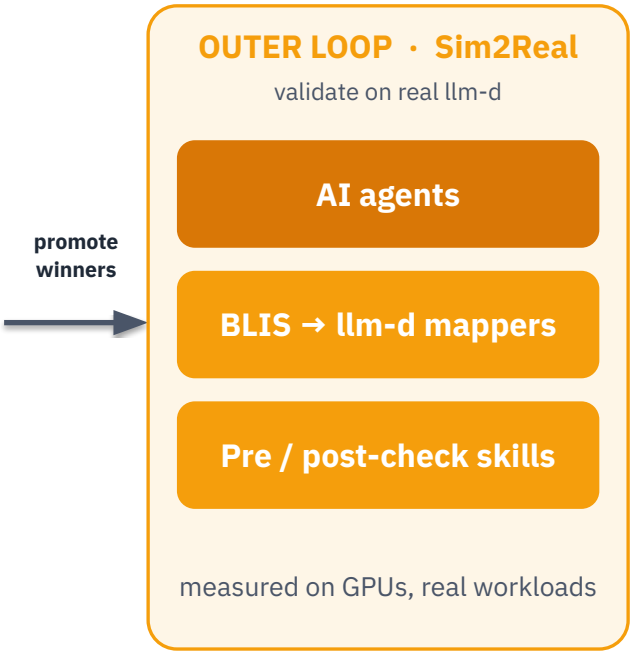


- CPU-based discrete event simulator for distributed LLM inference
- 200x faster: A workload that takes an hour on real hardware runs in seconds on BLIS

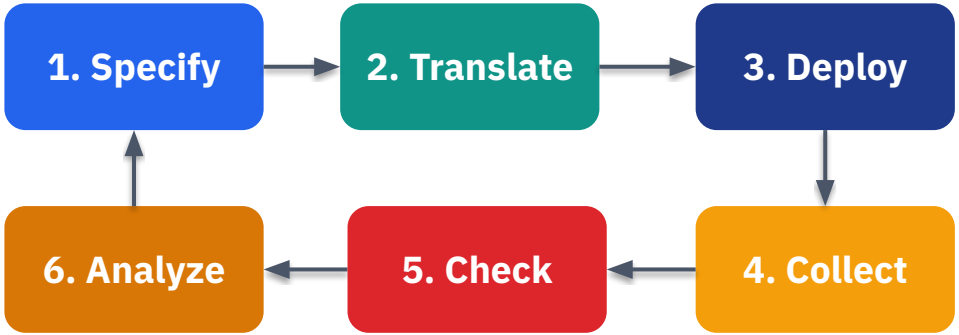
Evolve NOUS


INNER LOOP · discover in simulation
Plan
hypotheses, designs
Experiment
run policies in BLIS
Evolve
new policy candidates
Learn
findings, principles
target subsystem: admission control etc.

Validate



- **Expensive exploration already happened in simulation, at a fraction of the cost**
- Real hardware validate only the few candidates that survived the search
- **Simulation confirms direction; real hardware quantifies magnitude**



Inside Validate: agents, mappers, skills



expert writer reviewer

three-agent teams: build + test gate

Mapper: BLIS signals → llm-d signals (i.e. KV, queue depth, priority)

What broke, and the checks it led to

REAL INCIDENT		PRE / POST-CHECK SKILL
Misconfiguration priorities silently ignored	→	**config parity + EPP log proof**
Wrong mapping KV load off by 100×	→	**code proof for every signal**
Deployment drift baseline ≠ treatment stack	→	**variant isolation check**
Repeated execution 25 runs silently lost	→	**idempotent, resumable steps**
Noisy hardware GPU throttling	→	**thermal check drops bad runs**

From validated to merged upstream

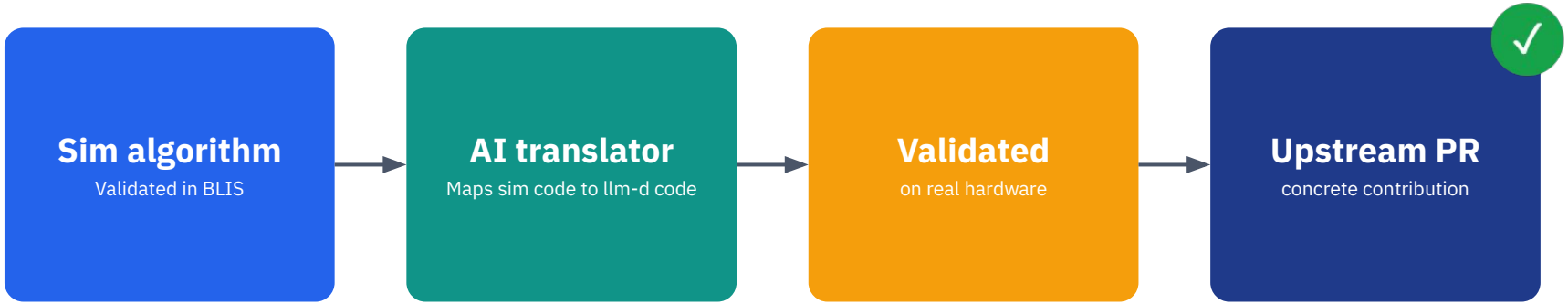


#959 admitter	unchecked parameters	→ **input validation + usage docs**
#1956 flow control	**starvation bug in lower bands**	→ **shared tick, contract updated**
#3392 LoRA routing	"what if requests starve?"	→ **extra H100 runs**

Review caught what sim and benchmarks missed

Ship

Contribute new strategies, algorithms, and policies upstream to llm-d



- **Not a prototype: the output passes the full build and test suite**
- Full provenance: discovered by experiment, under known conditions, with evidence, then translated
- Different from just asking LLM to “suggest an algorithm” because the results are based on simulation + benchmarked results

Concrete examples

✓ merged upstream

1

Probabilistic admitter

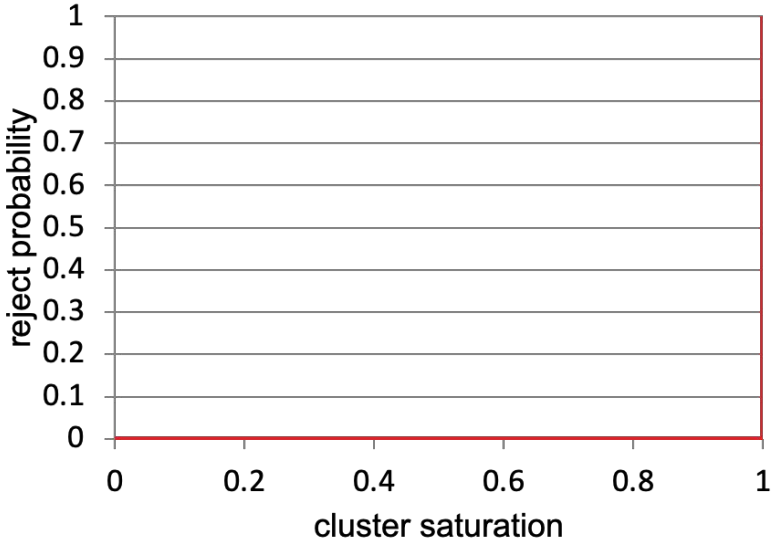
2

Soft-reflective flow control

3

Residency-aware LoRA routing

Observe: the admission binary cliff



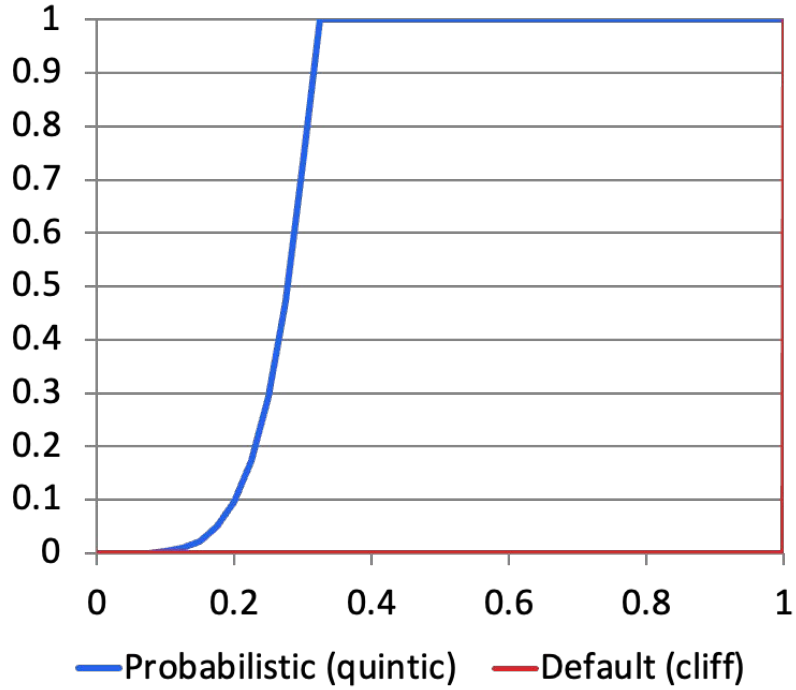
- Default behavior: **admit everything until saturation == 1.0, then hard-reject** all sheddable traffic
- `saturation = f(queue depth, KV-cache utilization)`
- By the time it fires, queues are deep and even protected traffic is very slow
- A binary cliff, not a graceful degradation

Discovered: probabilistic admitter

```
reject(droppable) = min(saturation^5 x 300, 1.0)
```

- Always admit protected traffic (priority >= 0)
- The 5th power and the 300 constant were discovered by Nous, not hand-tuned
- No explicit thresholds and protection ramps with stress

saturation	reject	behavior
0.12	~1%	Flows freely
0.25	~29%	Protection begins
0.34	100%	All droppable shed



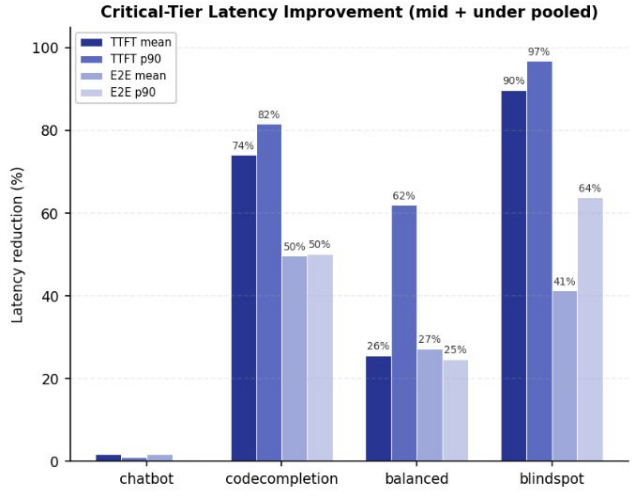
Results

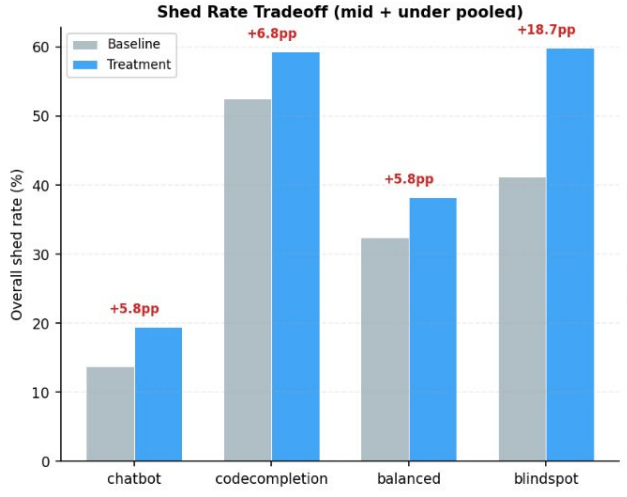
up to 97%

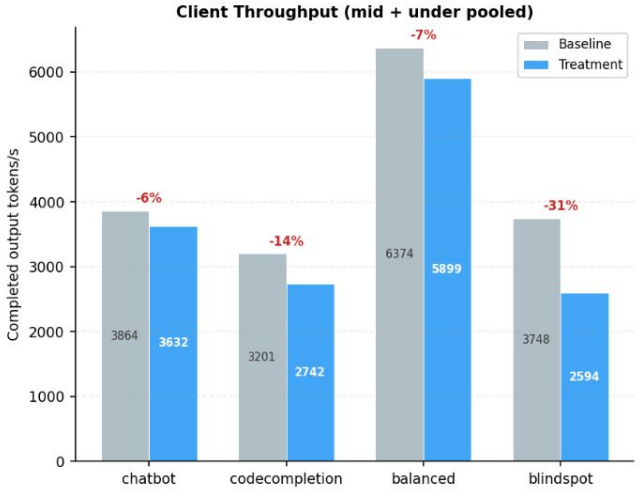
reduction in critical class TTFT p90

Set up

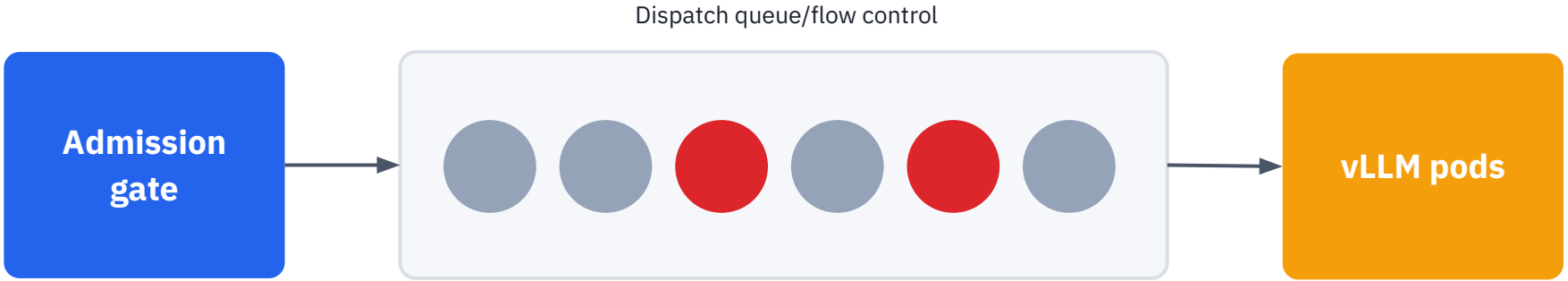
- 4xH100, Qwen3-14B on vLLM + llm-d
- 8 workloads (4 shapes x 2 loads)
- E2E latency 27-50% better in most
- Tradeoff: sheds 5-19 pts more low-priority traffic







Observe: critical queues behind sheddable



Critical requests (red) stuck behind sheddable (grey)

- Default static-usage-limit policy applies one threshold to every priority band
- Threshold 1.0 is effectively a no-op. Lowering it gates critical and sheddable at the same point
- **Under load, critical requests queue behind sheddable work**

Discovered: soft-reflective-ceiling

```
ceiling[0] = 1.0        # critical: never gated
ceiling[i] = 1 - i × saturation/(N-1)
```

- Two tiers collapse to a sheddable ceiling of 1 – saturation
- Above the ceiling it rate-limits dispatch (a proportional duty cycle), it does not hard-block
- **Zero free parameters: shape = band count + current saturation**
- Add a priority level and the reflection extends automatically

Saturation	Regime	Sheddable blocked	Critical blocked
0.50	Gating begins	0%	0%
0.60	Proportional	50%	0%
0.75	Proportional	67%	0%
0.90	Proportional	89%	0%
≥ 1.00	Hard block	100%	0%

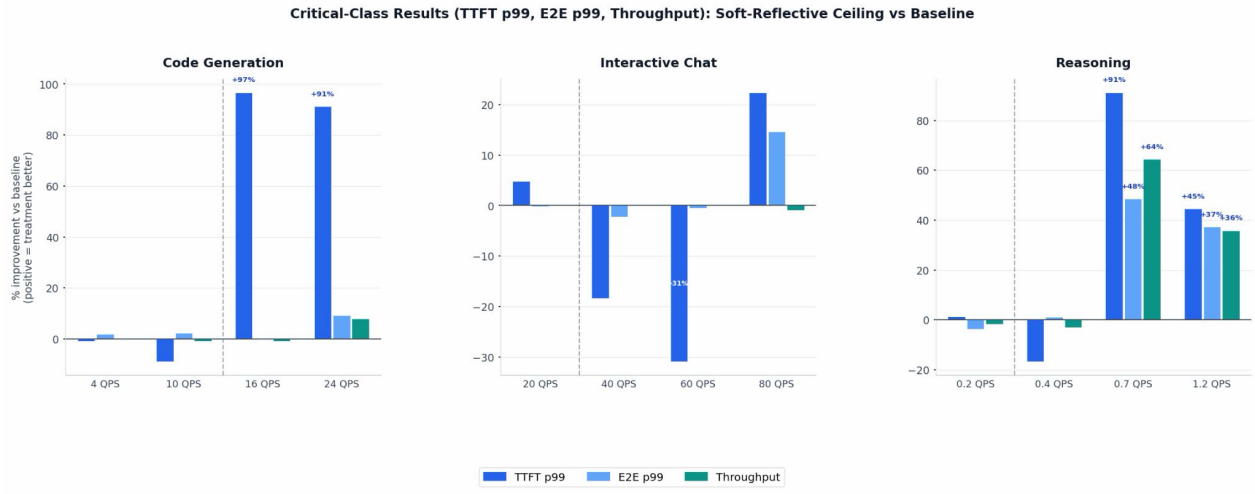
Results

up to 98%

reduction in critical class TTFT p99 at near capacity load

Set up

- 2xH100, Qwen3-14B on vLLM + llm-d
- Code gen, 16 QPS: 13.6s → 263 ms
- Reasoning, 0.7 QPS: 330s → 16s, throughput +64%
- Zero requests rejected: only reorders

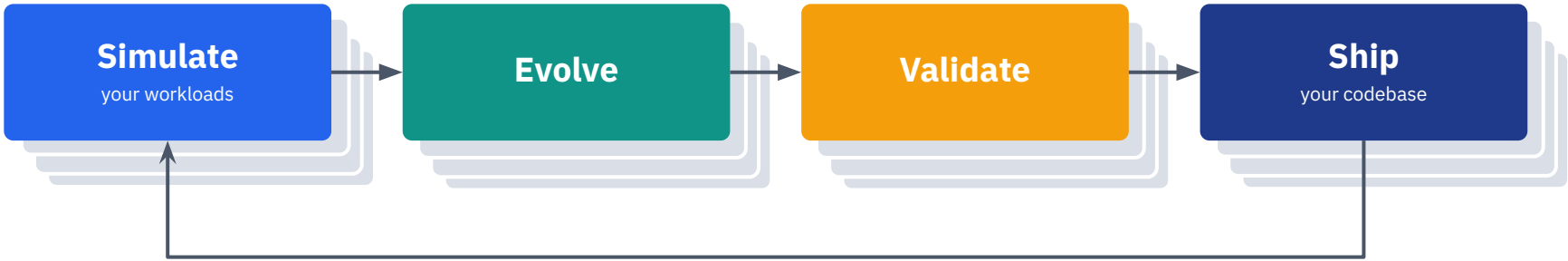


Shippable today with llm-d

- Three PRs (merged) in llm-d
  - https://github.com/llm-d/llm-d-router/pull/959
  - https://github.com/llm-d/llm-d-router/pull/1956
  - https://github.com/llm-d/llm-d-router/pull/3392
- Enable via single YAML change, no new infra
- Ships with simulation-suggested defaults that work out of the box

```
flowControl:
  usageLimitPolicyPluginRef: soft-reflective-ceiling-policy
plugins:
  -    type: soft-reflective-ceiling-policy
  -    type: queue-scorer
  -    type: kv-cache-utilization-scorer
```

Advancing the distributed inference at speed!



The loop is a development methodology, not an llm-d-only tool

- **Turn exploratory ideas into testable hypotheses**
- Evaluate SLO behavior before you spend money on hardware for discovery and testing
- Reserve hardware for validation, not discovery

Simulate → Evolve → Validate → Ship

KubeCon CloudNativeCon North America 2026

llm-d	https://llm-d.ai/
BLIS (simulator)	https://llm-d-incubation.github.io/llm-d-perf-simulator/
Nous (agentic evolution)	https://github.com/AI-native-Systems-Research/agentic-strategy-evolution
Blog series	https://ai-native-systems-research.github.io/ai-native-systems-research/blog/category/llm-d/

Thank you! Questions?

jing.chen2@ibm.com | toslali@ibm.com

