OLYMPIA AI DATA INFRASTRUCTURE SUMMIT 2026

Aalto University

State Management for Agentic AI Systems

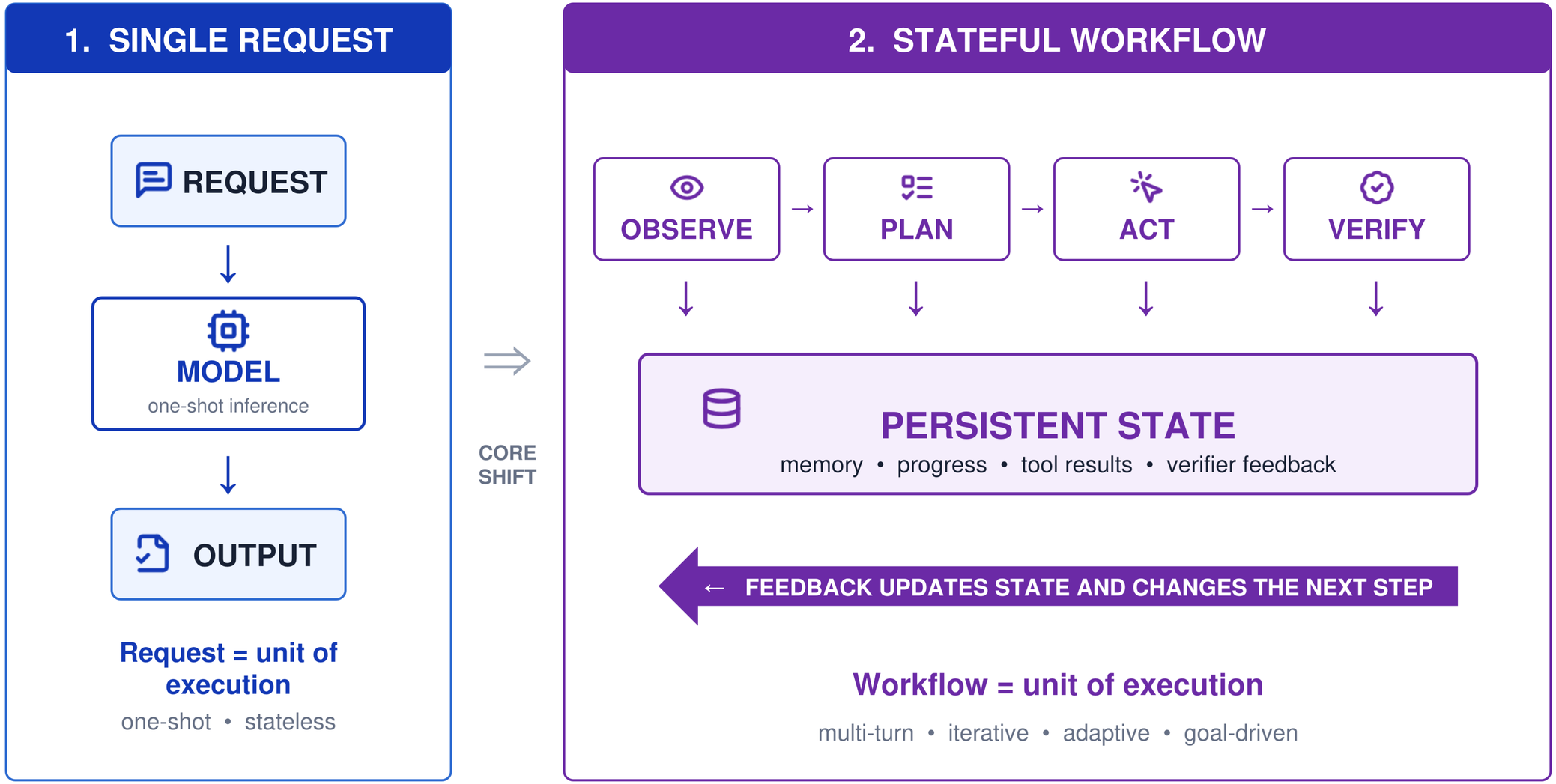
From RL Post-Training to Agentic Workflows

Olympia Summit · Heraklion · 4 November 2026

Bo Zhao
bo.zhao@aalto.fi **https://zbjob.github.io/**
Assistant Professor, CS Department, Aalto University

Why do agentic workflows need state management?

State persists across steps and changes what should execute next



State persists across steps—and must shape what executes next.

Where does state enter the feedback loop?

C1–C5 link workflow, resources, verification, experience and learning

C5: Training State

Parallelism and state reconfiguration.

RL Post-training Update (Policy Update)

1 Agent / Policy

Decide next action conditioned on context and past outcomes.

C1: Workflow State

Long-horizon, branching tasks.

5 Measurements / Error Signals / Rewards

Collect outcomes, error signals, and rewards.

Synthesize quantum circuits for scientific & engineering tasks

Quantum Chemistry | Drug Discovery | Portfolio Optimization

Generate → Simulate → Execute → Measure → Learn

2 Program / Workflow

</>

Generate quantum circuits / programs (e.g., OpenQASM) and tool calls.

C4: Experience State

Feedback becomes the data plane.

C2: Adaptive Resources

Reallocate compute as load shifts.

4 QPU Execution & Measurement

Execute selected circuits on real quantum hardware and collect measurements.

3 HPC Simulation & Quantum Job Preparation

Simulate circuits, transpile, prepare QPU jobs, and compute verifier signals.

C3: Tool / Verifier State

Costly tool calls (simulators, compilers, verifiers, QPUs) define correctness.

Executed on classical systems (HPC)

Executed on quantum systems (QPU)

LUMI Supercomputer

TOP500 Top-10 supercomputer



- 11,912 AMD MI250X GPUs
- 4,096 AMD EPYC CPUs
- HPE Cray EX
- High-bandwidth, liquid-cooled HPC system

VTT Q50 Quantum Computer



- 53 qubits (superconducting)
- Cryogenic execution
- Real measurement outcomes
- Integrated with LUMI

The state wall

How should a runtime respond when state changes?

Fixed execution

Single model calls

Model-centric execution

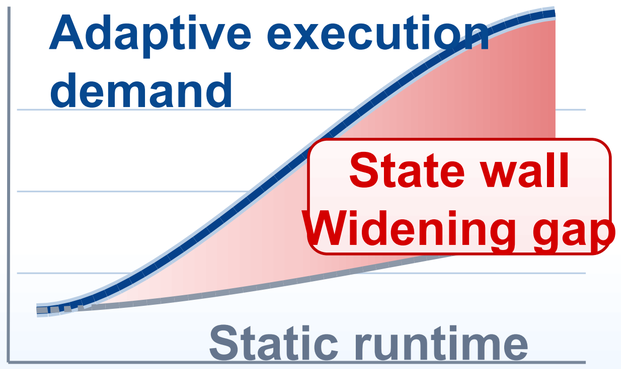
Input → Model → Output

Passive state
Preset execution

Goal: faster model calls

The state wall

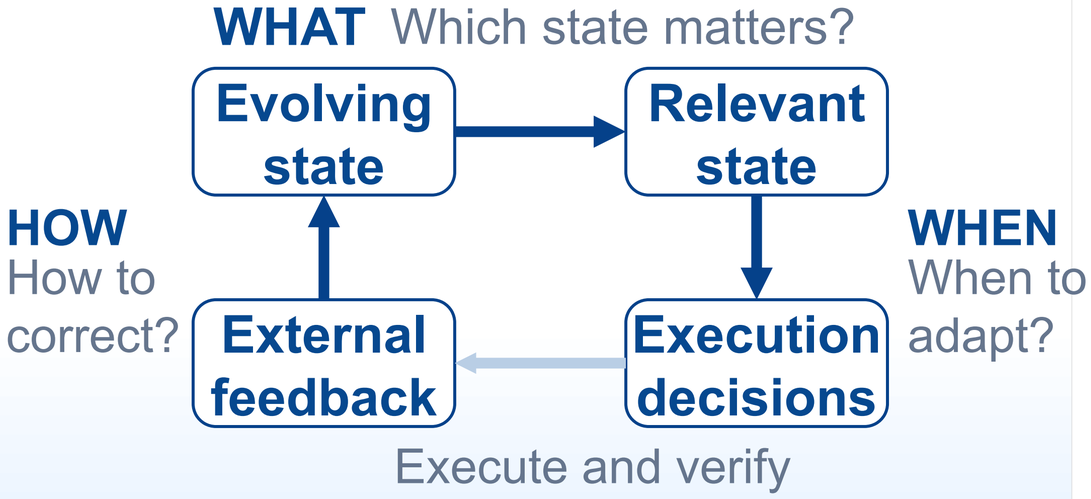
State evolves faster than runtimes adapt



Workflow complexity

Adaptive execution

Execution adapts as state evolves



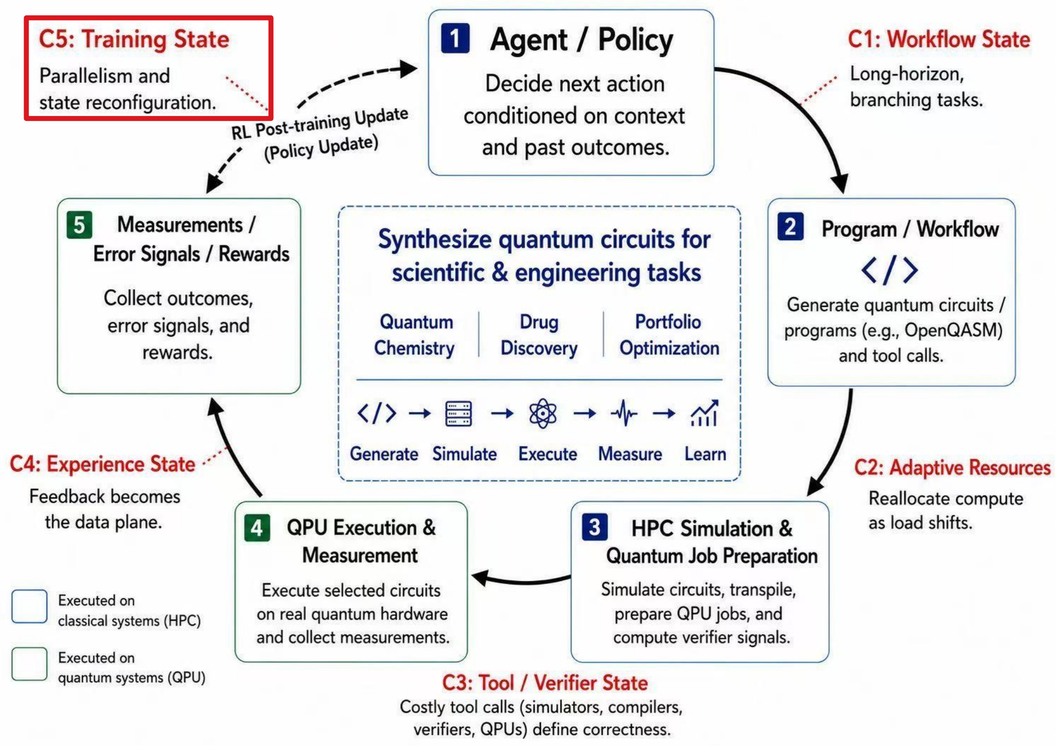
Goal: sustain correct execution

State becomes a first-class input to execution decisions

EXPOSE: MSRL makes RL computation schedulable

MindSpore Reinforcement · USENIX ATC 2023 · Joint research with Huawei

Location in the research map



C5 · Training state

RL computation and data flow

↓ **Expose computation as fragments**

Distributed execution

Map fragments through distribution policies

The same RL algorithm can use different distribution strategies.

MSRL: how can RL computation span devices?

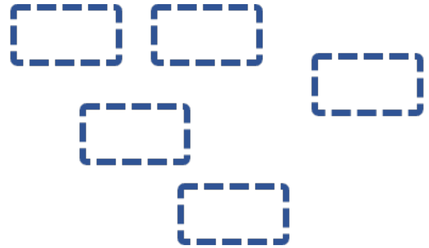


Dataflow fragments separate computation from placement

MindSpore Reinforcement · Joint research with Huawei

1. Express RL computation as **dataflow graphs**
2. Execute computation on **heterogeneous devices**
3. Support **policies** for **parallelisation/ distribution**

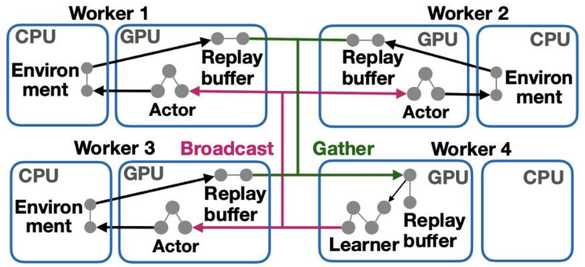
Fragmented Dataflow Graph (FDG)



Fragments

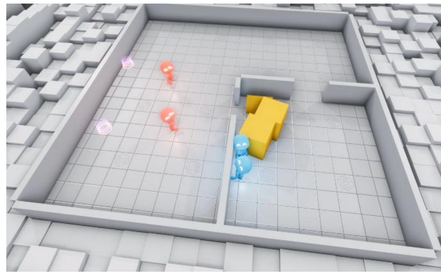


Distribution Policies

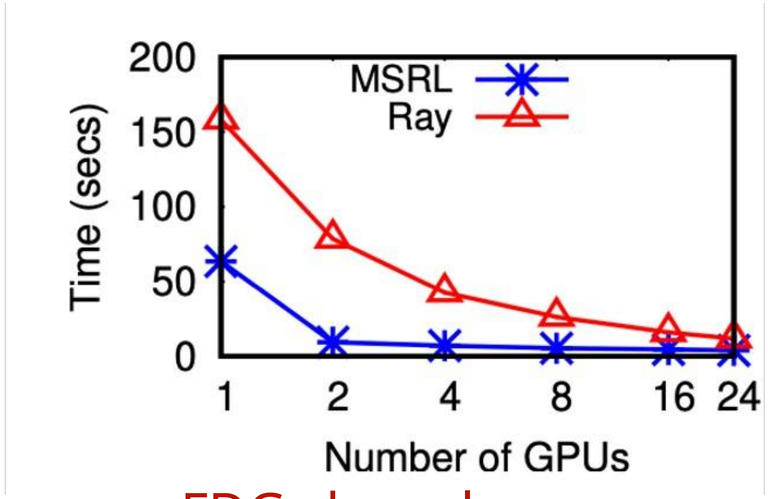


MSRL: Distributed Reinforcement Learning with Dataflow Fragments (USENIX ATC'23)

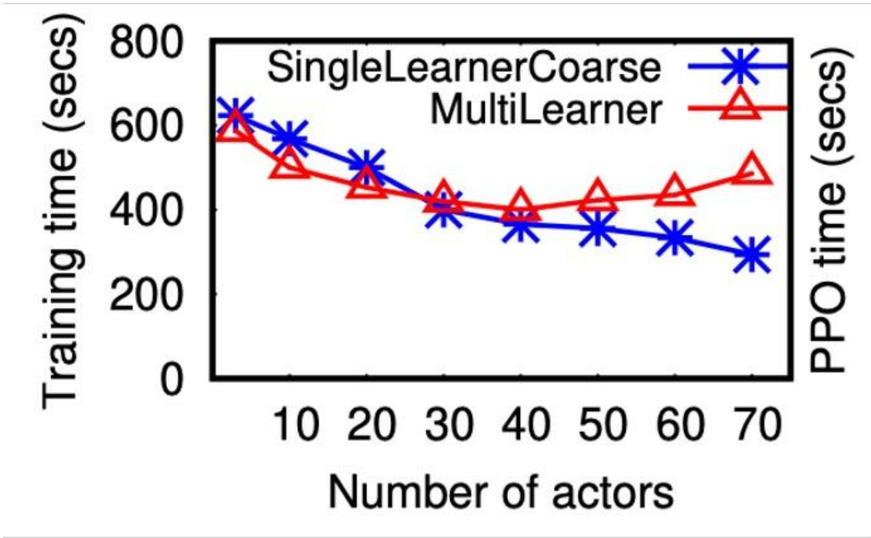
MSRL: can flexible execution remain fast?

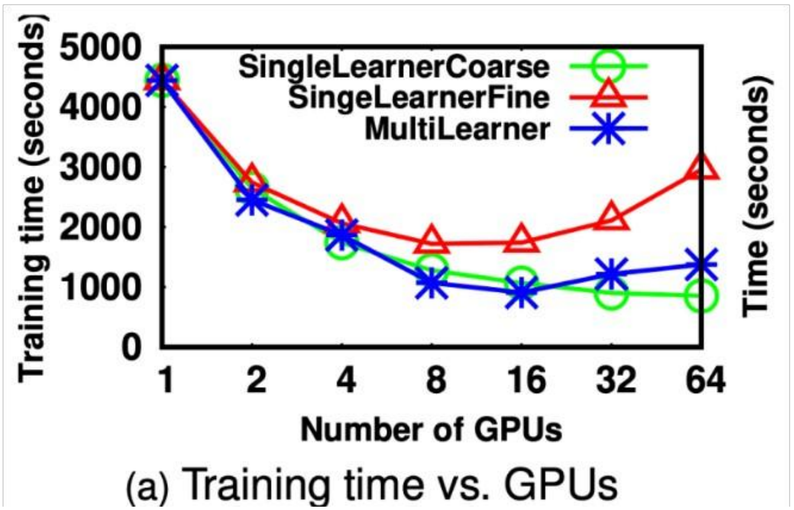


MindSpore Reinforcement combines performance with flexible policies



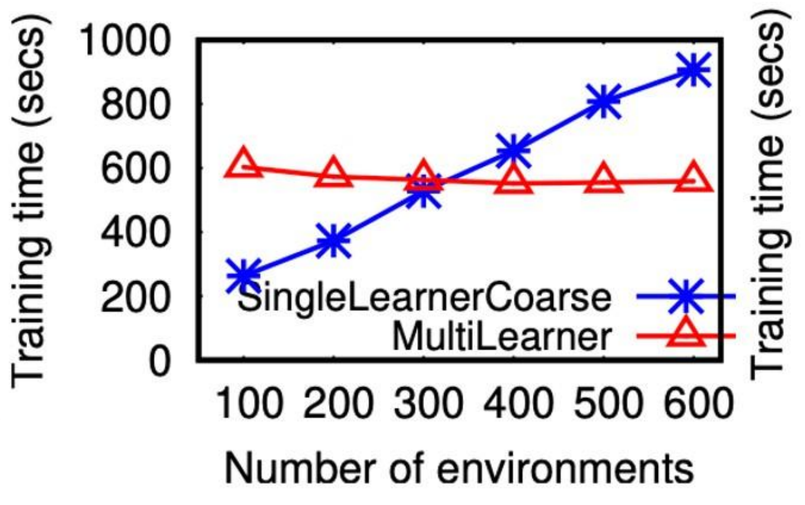
FDGs have low overhead





(a) Training time vs. GPUs

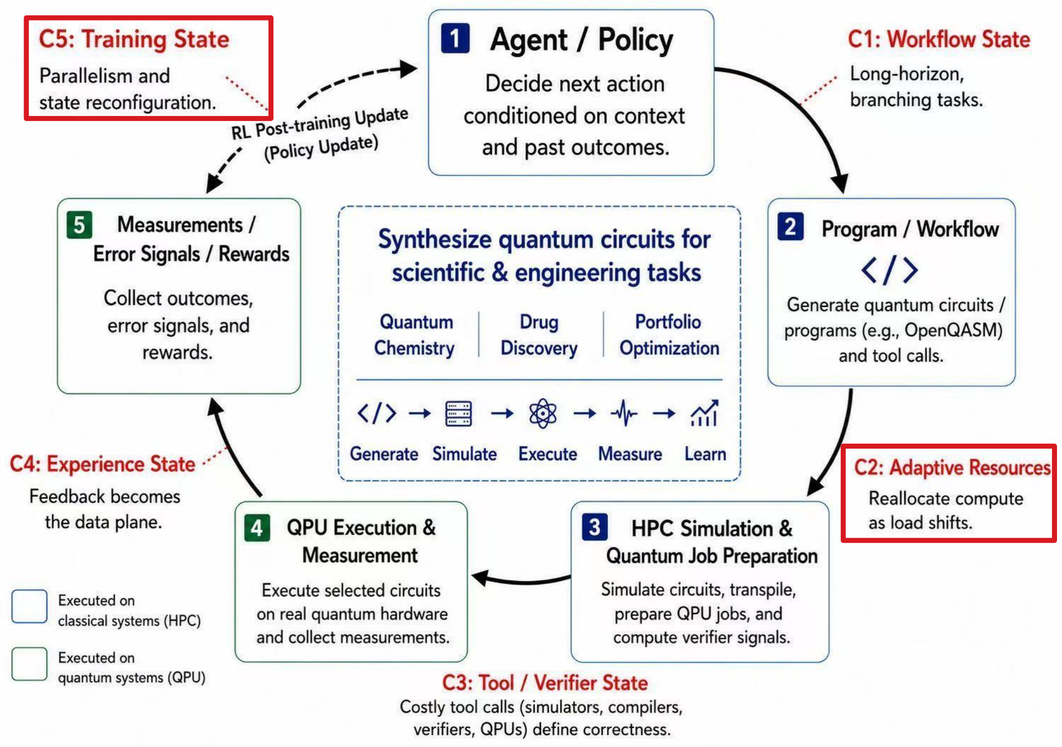
No single best distribution policy



TRANSFORM: Tenplex preserves state as resources change

SOSP 2024

Location in the research map



C2 · Adaptive resources

The number of available devices changes

↓ **Transform the state layout**

C5 · Training state

Preserve logical tensors and data access

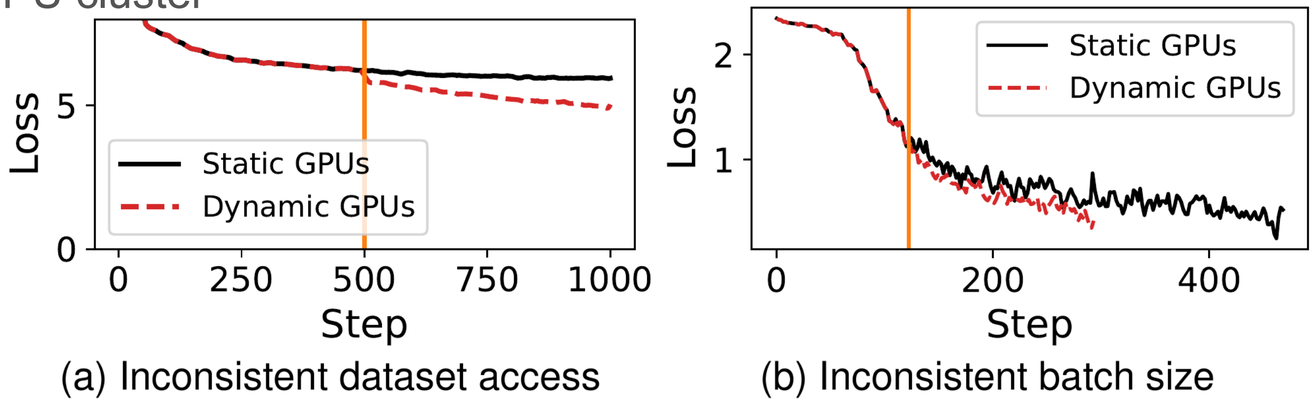
Change physical placement while preserving logical training state.

Why is changing GPUs a state problem?

A new layout can change data access, batch size and performance

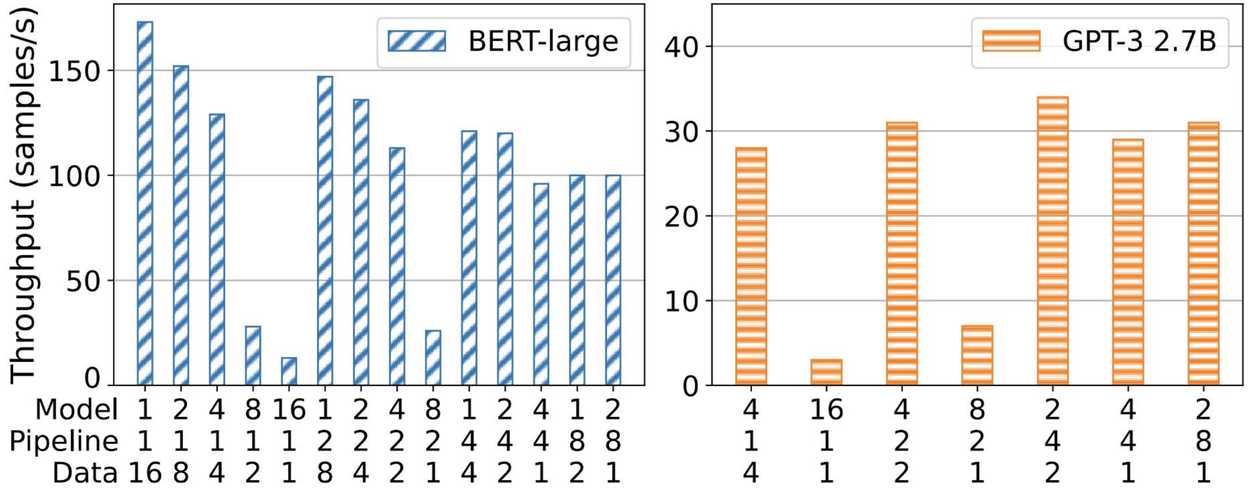
BERT-large and GPT-3 2.7B on a 16-GPU cluster

- Training consistency



(a) Inconsistent dataset access (b) Inconsistent batch size

- Training performance



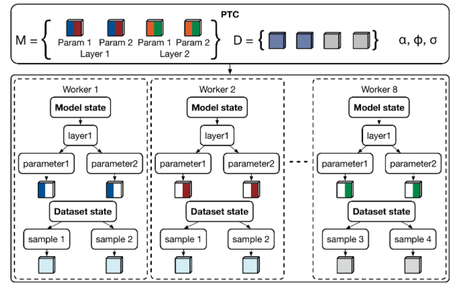


Tenplex: how can state follow changing resources?

Tensor collections expose state that the runtime can transform

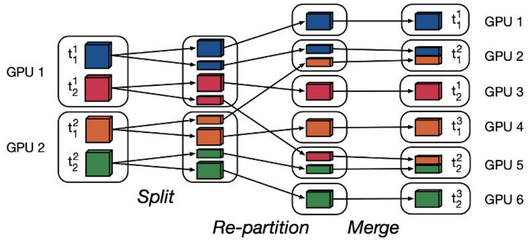
1. Capture state as **tensor collections**

Parallel tensor collections (PTC)



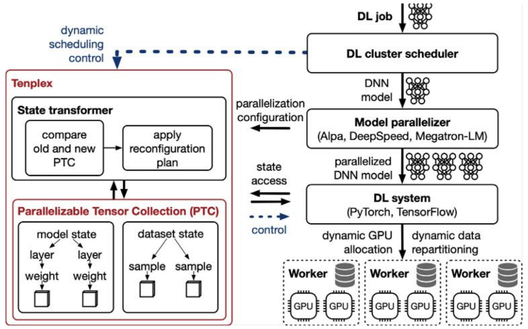
2. Transform PTCs with **multi-denominational parallelism**

Transformation algorithm



3. **Integrate with existing DL engines**

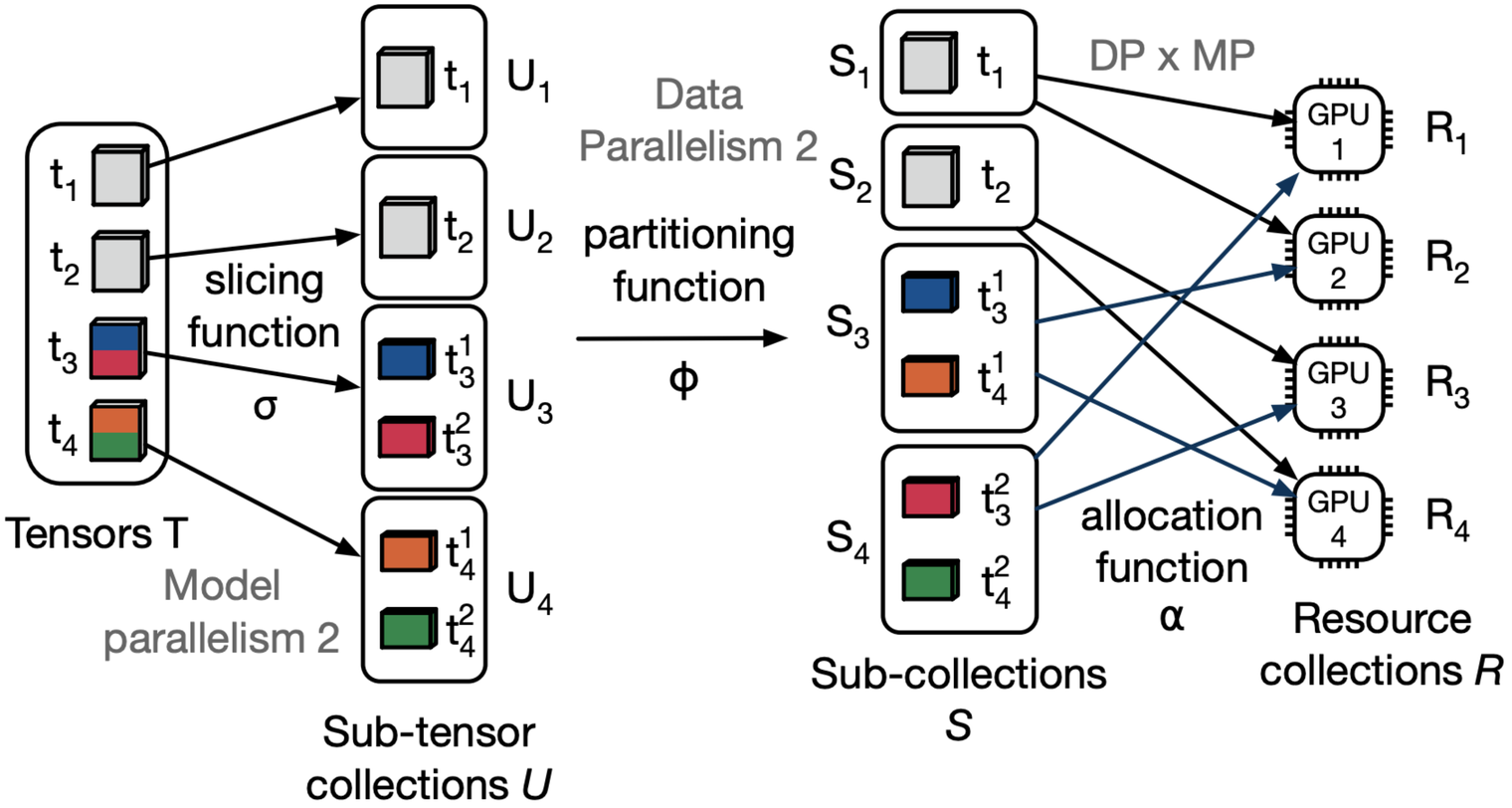
Virtual file system and in-memory data store



TENPLEX: Changing Resources of Deep Learning Jobs using Parallelizable Tensor Collections (SOSP'24)

Tenplex: what stays fixed when the layout changes?

Logical tensors persist while slicing, partitioning and placement change



PTC = (T, σ, φ, α)

T: model and data tensors

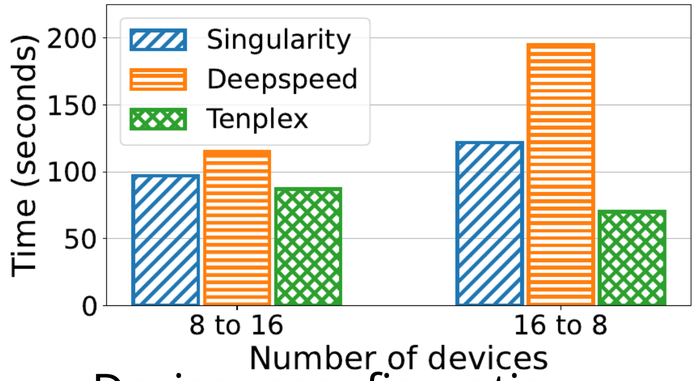
σ: slice tensors

φ: partition slices

α: map to GPUs

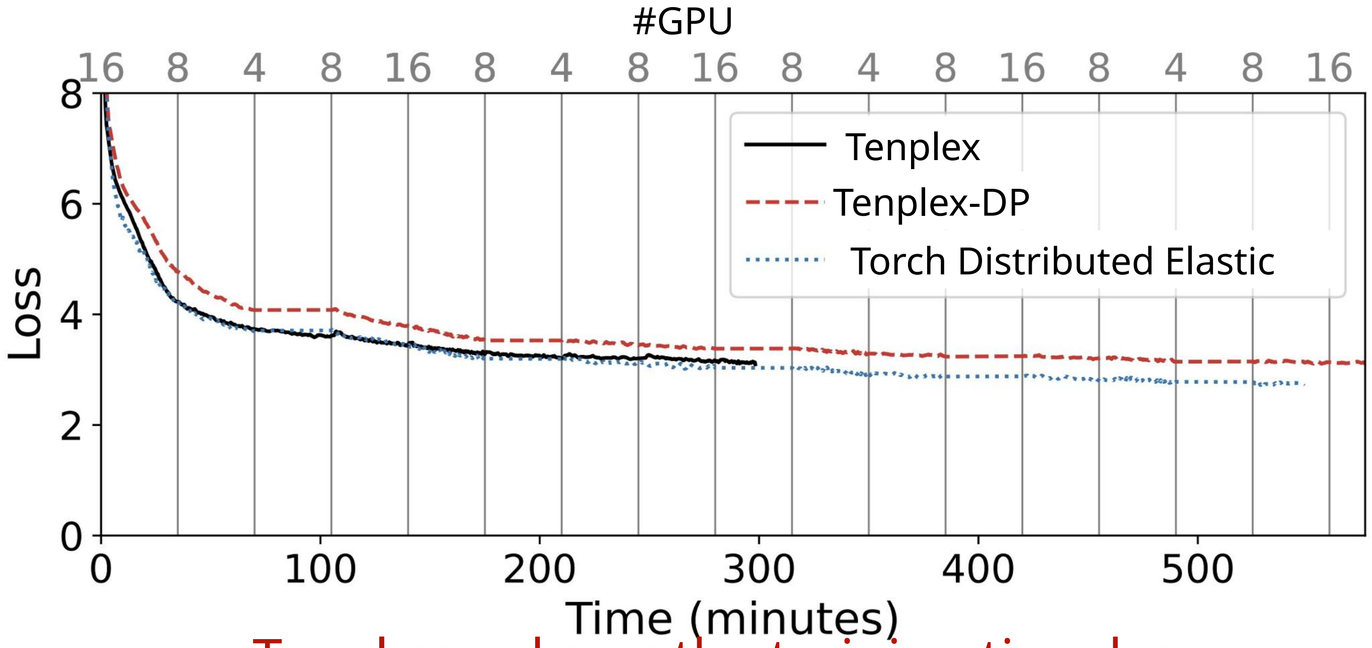
Transform the layout, preserve logical state.

Tenplex: faster training under changing resources

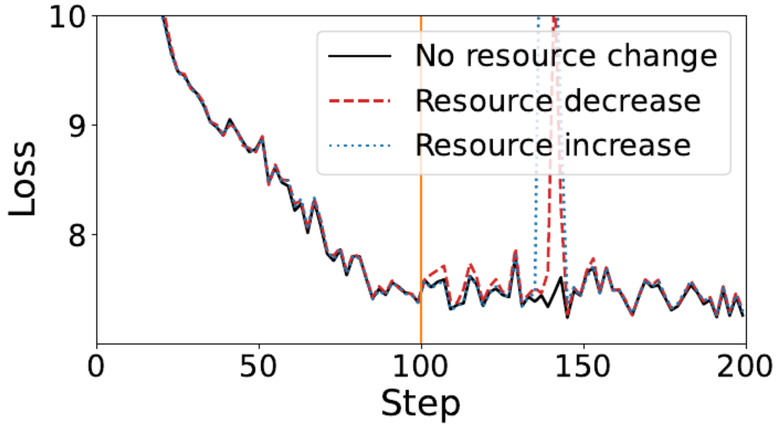


Device reconfiguration time

PTCs have low overhead

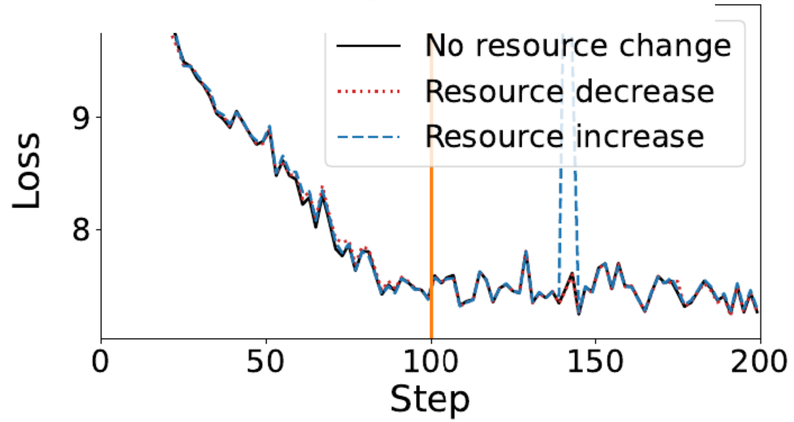


Tenplex reduces the training time by 46%



(c) Model parallelism

Changing devices do not affect training consistency



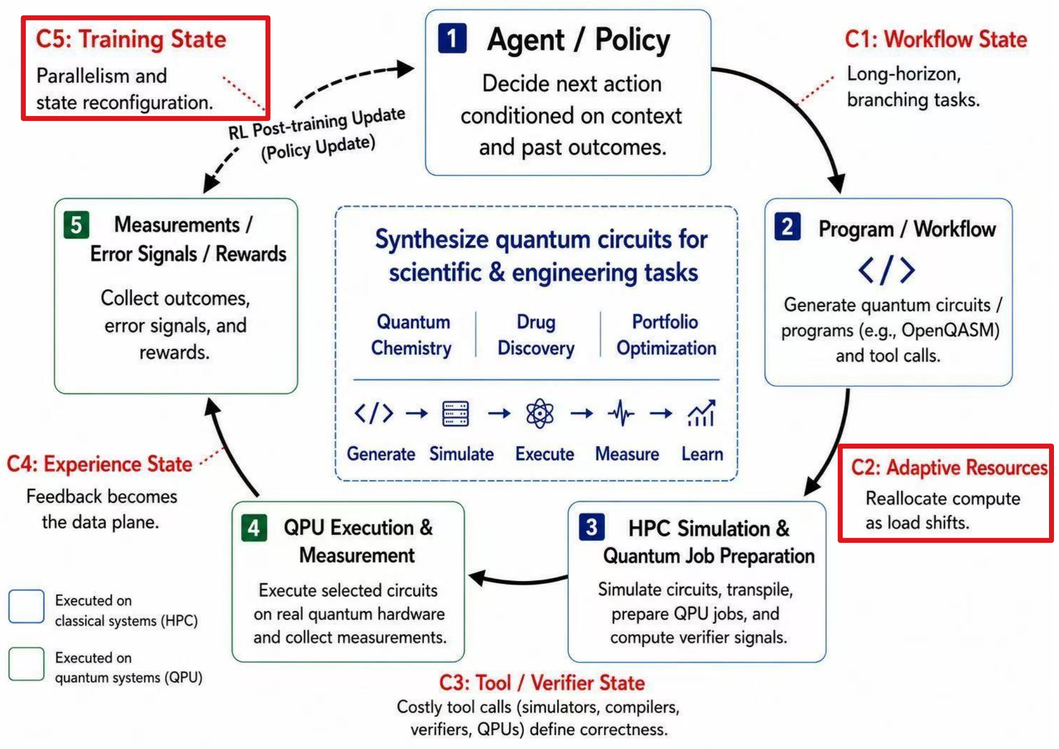
(b) Pipeline parallelism

COORDINATE: Nereus aligns model stages and resources

Nereus · arXiv preprint, 2026

Focus 1 of 2

Location in the research map



C5 · Training state

Model stages have different execution needs

↓ **Weigh switching benefits against costs**

C2 · Adaptive resources

Coordinate allocation with stage transitions

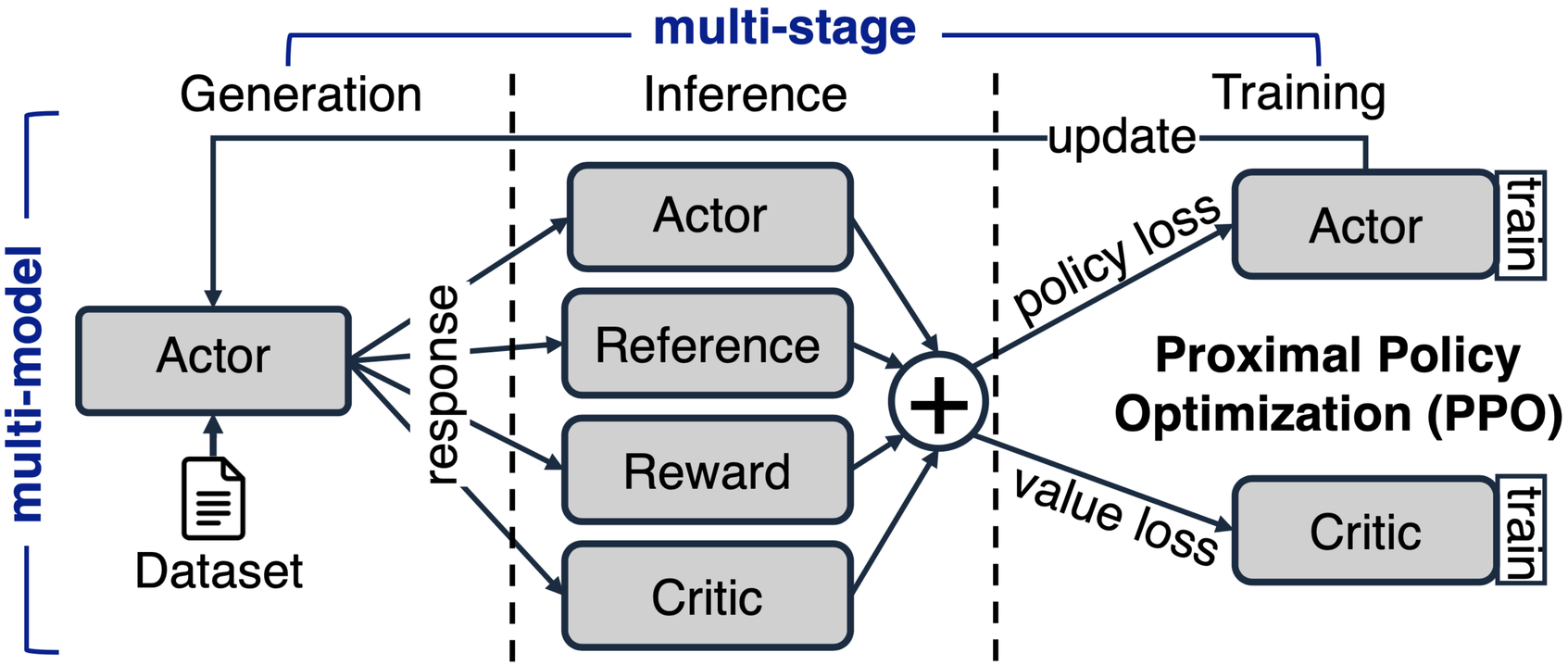
State across model stages guides resource reconfiguration.

Why must post-training stages adapt together?

A faster stage can move the bottleneck to the next stage

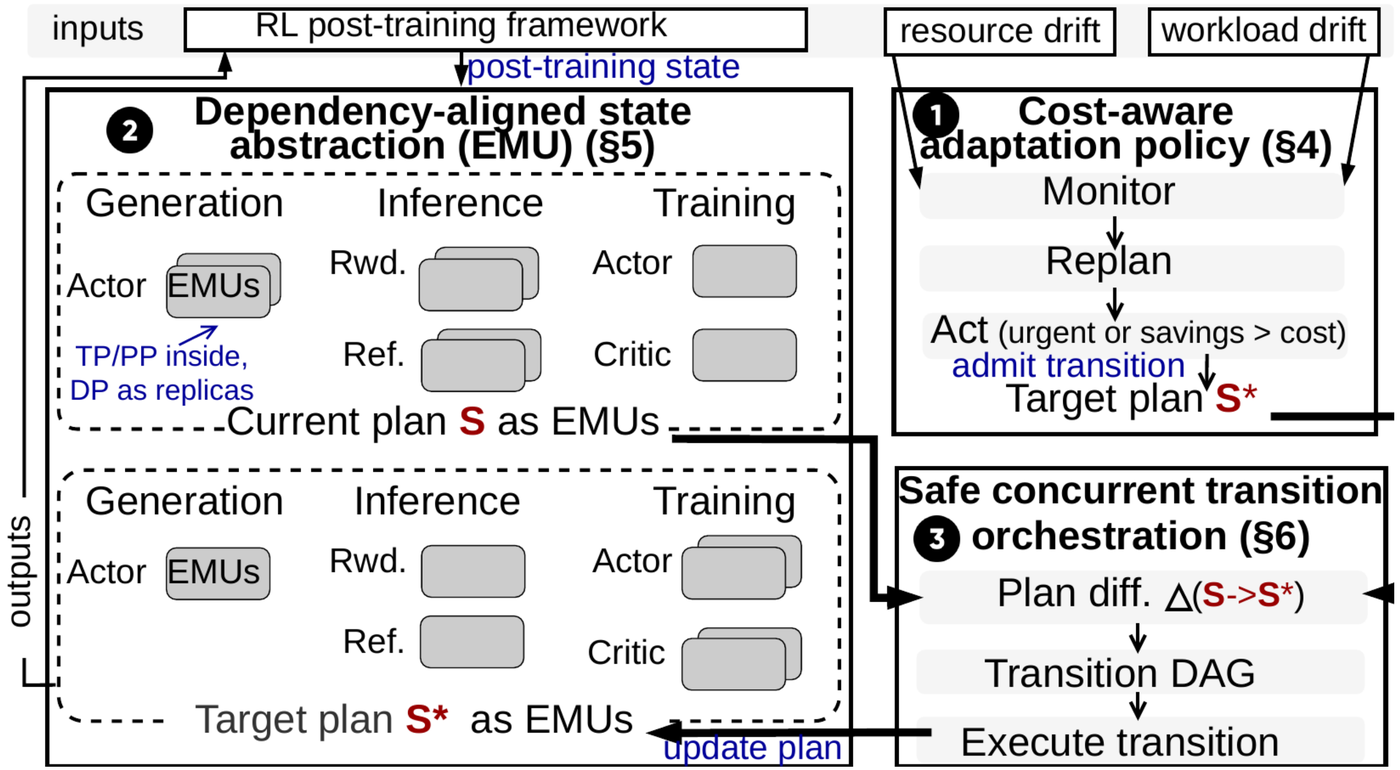
Models span generation, inference, evaluation, and training.

State dependencies couple local and global change.



Nereus: how do model stages adapt together?

One global plan coordinates stage layouts and state transitions



What changes?

Model-stage state, represented by EMUs.

When to switch?

Urgent change, or gains that justify transition cost.

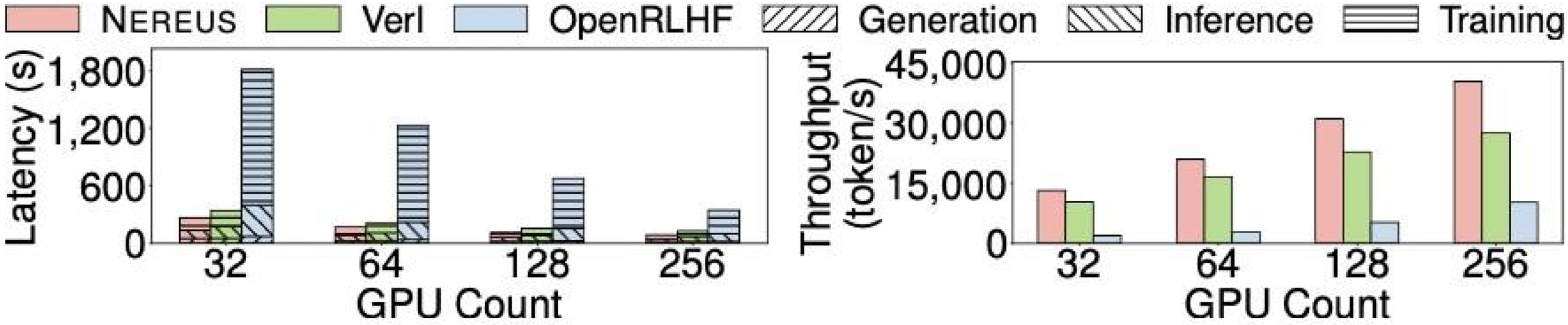
How to switch?

Coordinate state transitions through dependency DAGs.

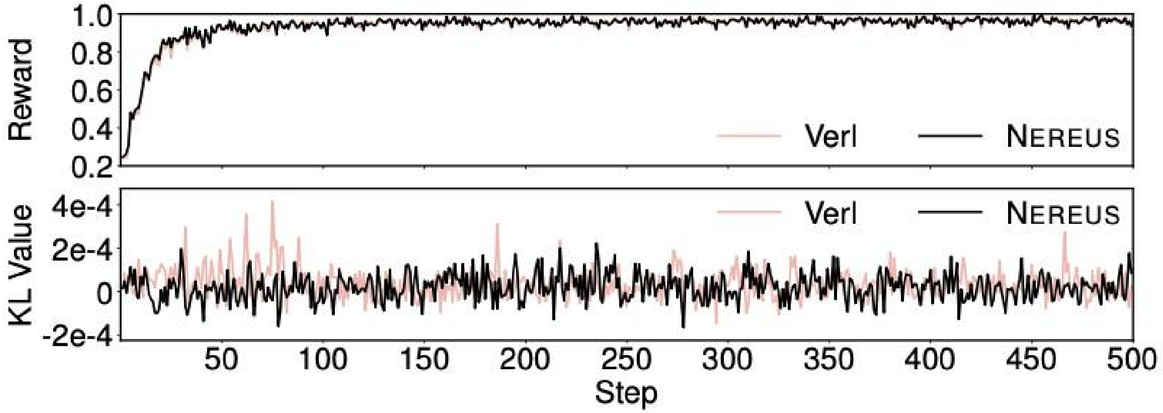
Nereus, arXiv:2609.34645v1 · Figure 1, Section 5.1

Nereus: higher throughput than Verl and OpenRLHF

Does coordination improve speed while preserving learning?



Lower iteration latency



Higher throughput

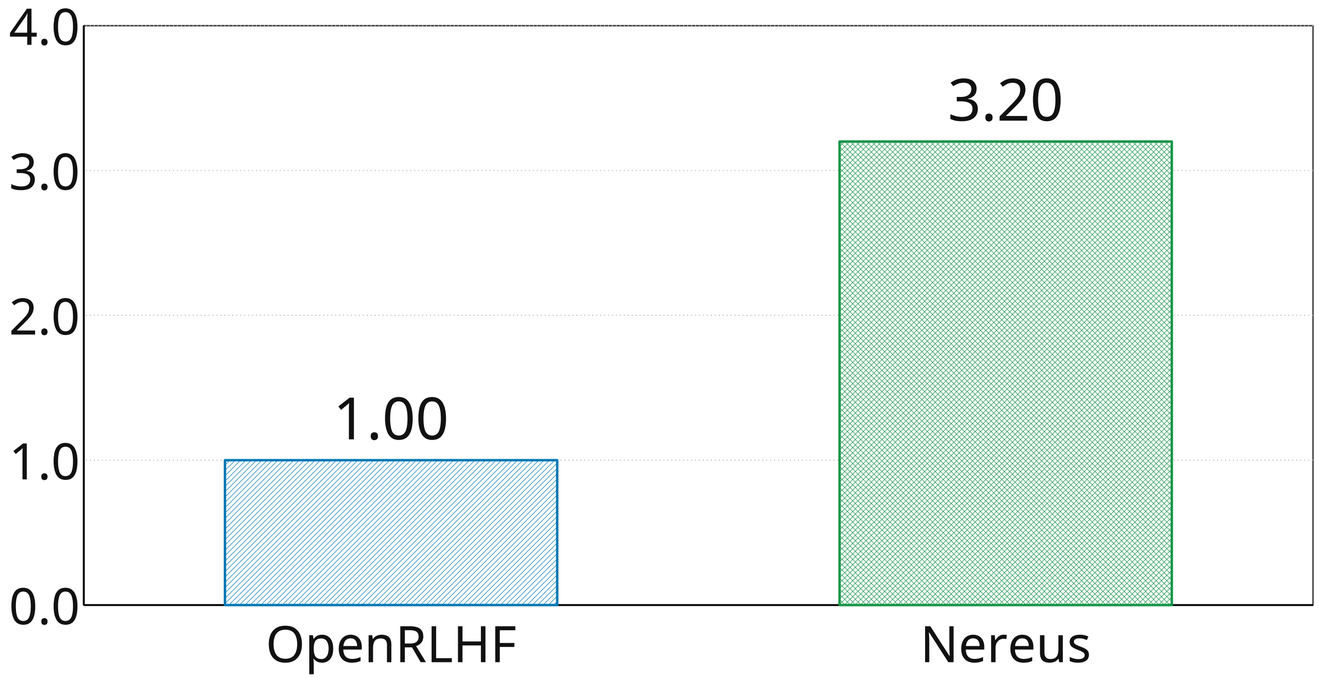
1.47× throughput vs Verl at 256 GPUs

Similar reward and KL over 500 steps

Nereus: 3.20× throughput at 1,024 GPUs

Llama-3.1-8B PPO, 128 AMD nodes × 8 MI250X GPU dies

Throughput relative to OpenRLHF



Same workload and 1,024-GPU budget

Light state transition overhead

Six transitions
from 32 to 1,024 GPUs

49.5 s (transition time) /
62,353 s (total exe time)

0.079% of total run time

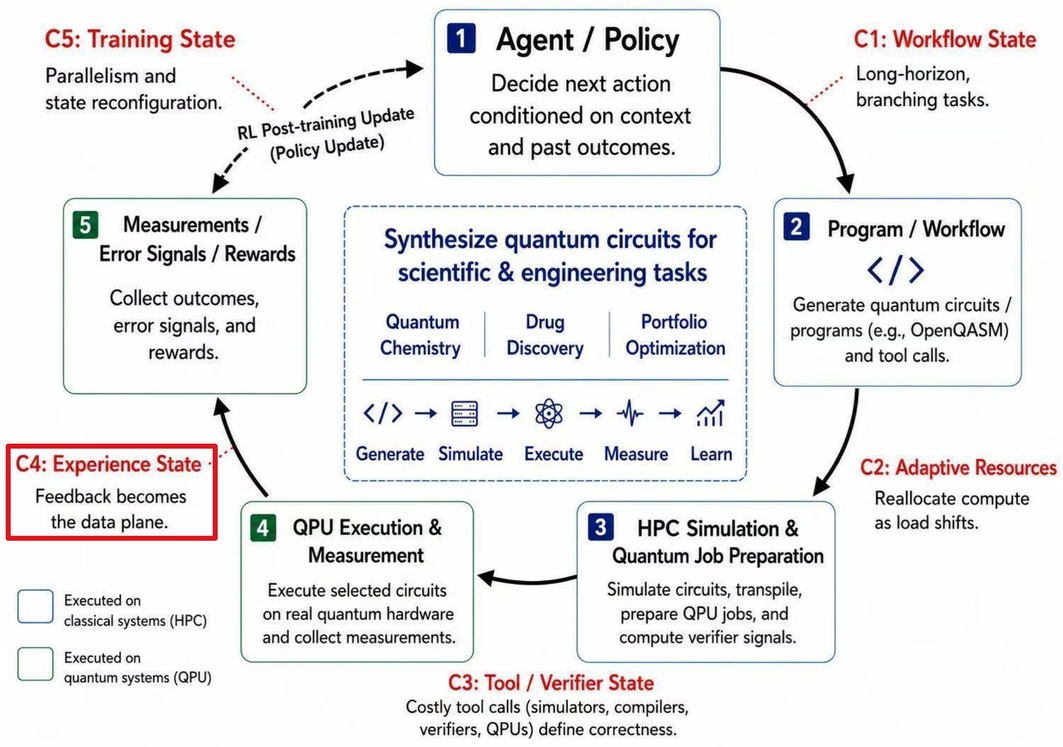
Separate 1,000-step PPO run
reaching 1,024 GPUs

Nereus, arXiv:2609.34645v1 · Figure 8 and Section 7.4.1 · Throughput and transition cost are separate runs

DELIVER: Conduit keeps experience moving

Conduit · arXiv preprint, 2026

Location in the research map



C4 · Experience state

Rollouts must reach the right learners

↓ **Place and schedule experience**

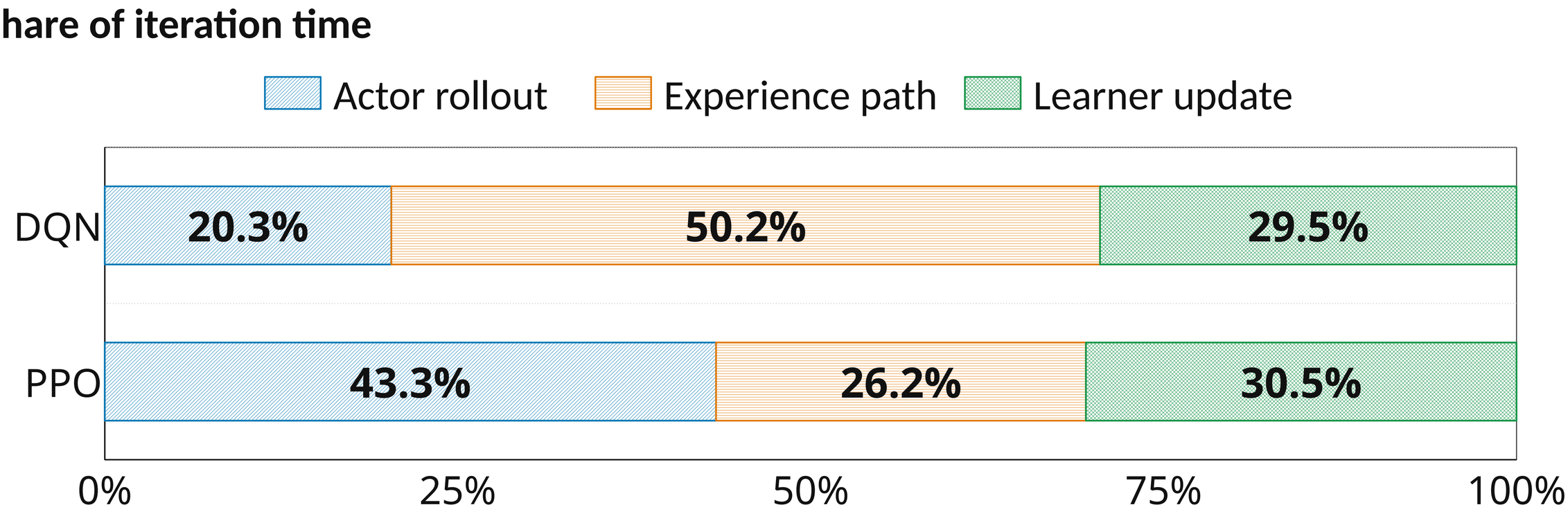
Timely delivery

Overlap data handling with computation

Timely experience delivery keeps rollout and learning moving.

What can keep rollout and learning waiting?

RLlib, eight actors and eight learners on eight A100 GPUs



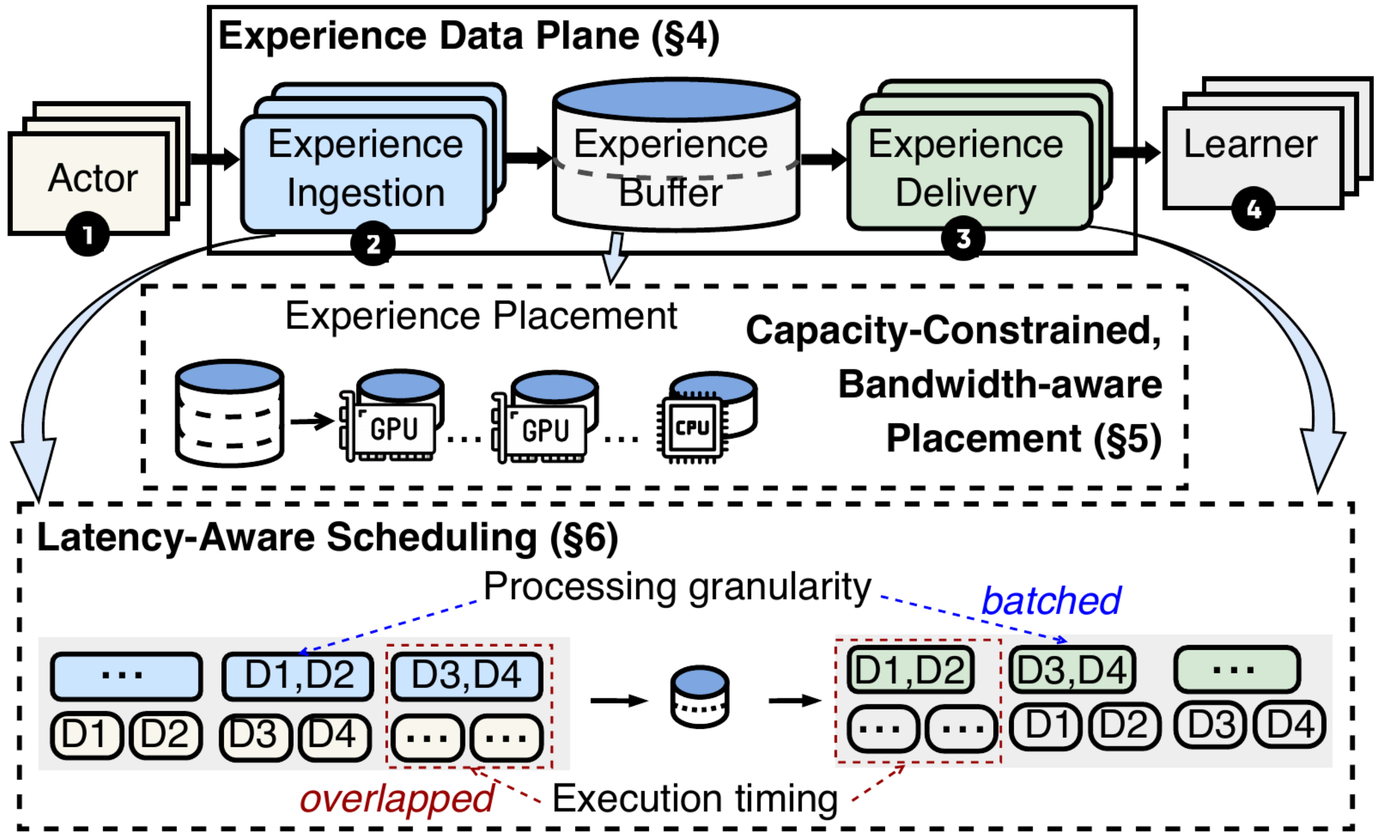
Experience handling takes 26–50% of iteration time

Handling operations per iteration: 33 for DQN, 27 for PPO

Conduit, arXiv:2609.24456v1 · Figures 1–2

Conduit: where should experience live and move?

Placement and scheduling overlap data handling with computation



Compute adapts.
Data must follow.

Place data by memory capacity and bandwidth.

Overlap data handling with rollout and learning.

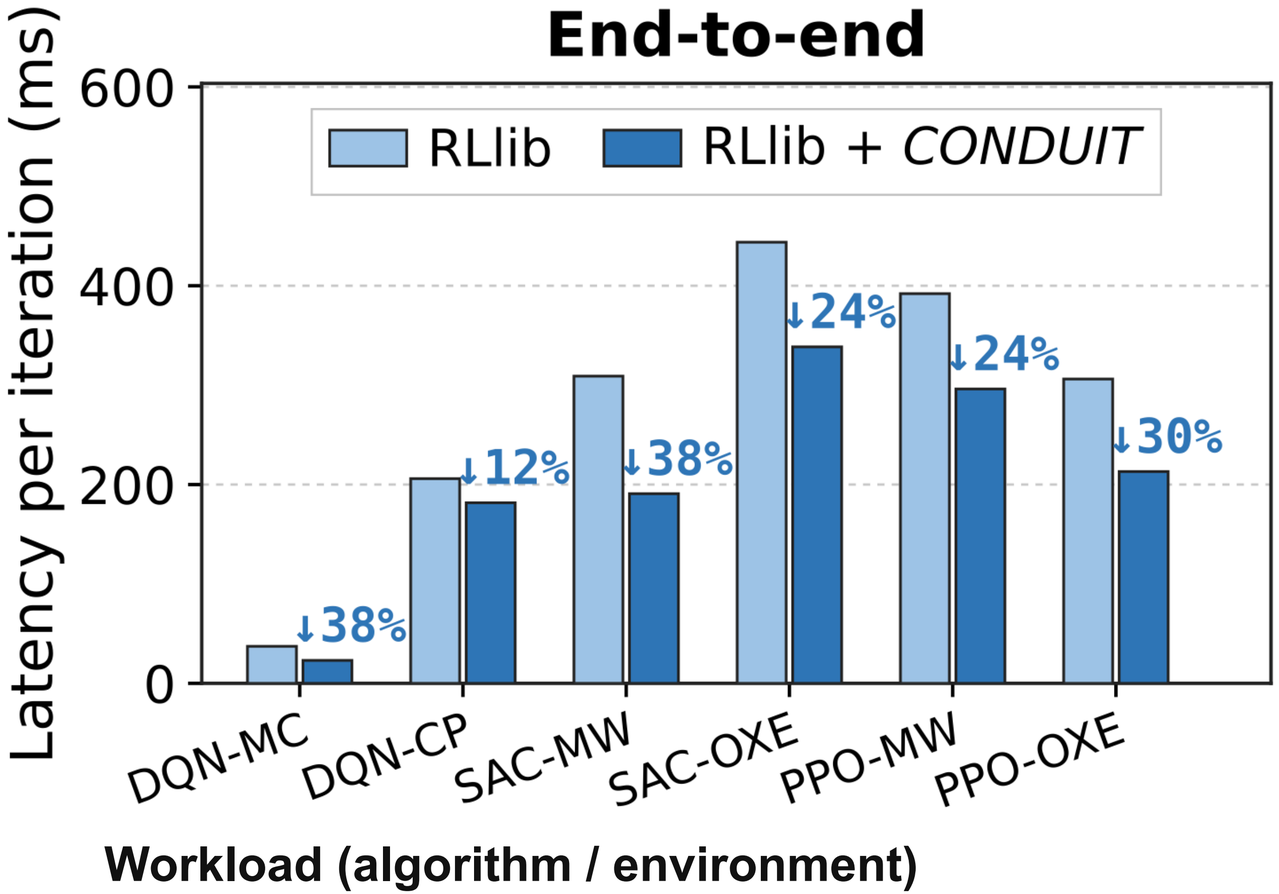
Preserve freshness and replay semantics.

Conduit, arXiv:2609.24456v1 · Figure 7, Sections 4–6

Conduit: end-to-end gains and scaling

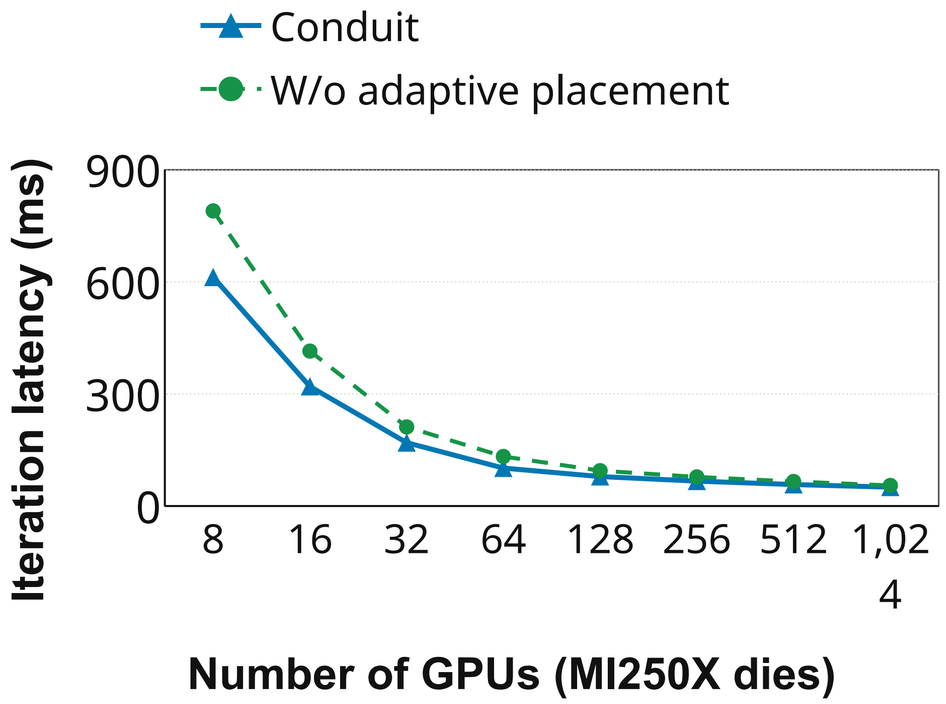
Baseline comparison and a separate fixed-workload scaling experiment

12–38% lower iteration latency



MC: MountainCar · CP: multi-agent CartPole
MW: Meta-World · OXE: Open X-Embodiment

Strong scaling to 1,024 GPUs



Fixed total batch: 32,768

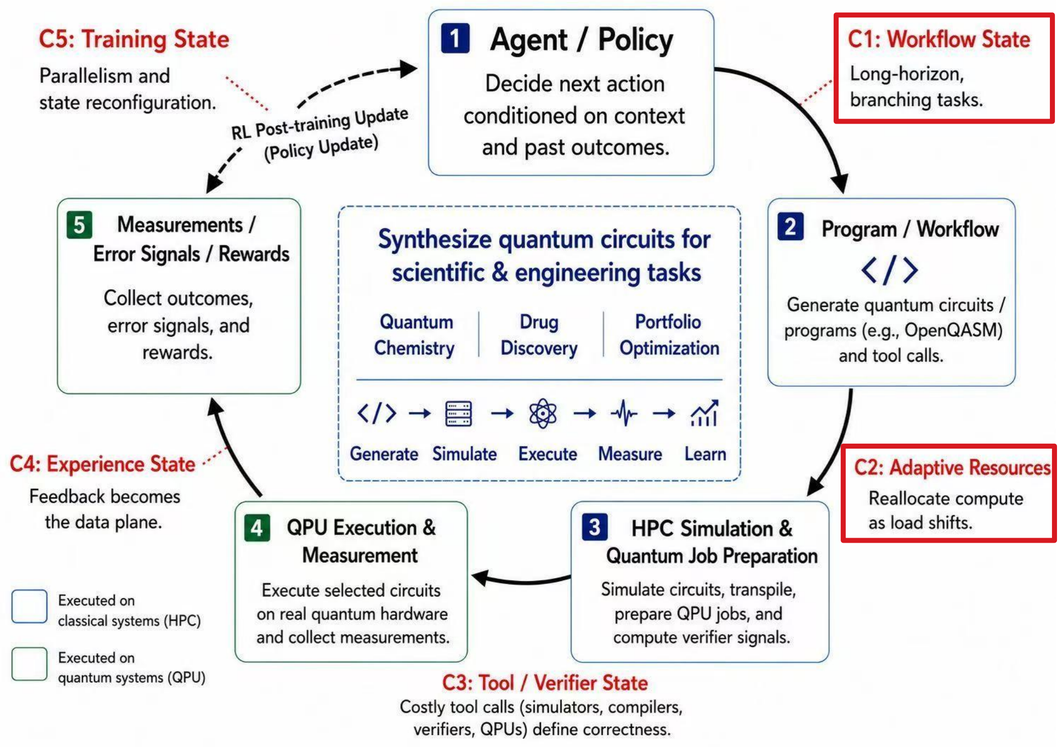
≈50 ms at 1,024 GPUs

Conduit · Fig. 11 (original panel), Fig. 16(b)
Separate experiments. Scaling values are approximate.

REUSE: sServe uses workflow and backend state

Work in progress

Location in the research map



C1 · Workflow state

Dependencies determine which work is ready

↓ **Combine with backend state**

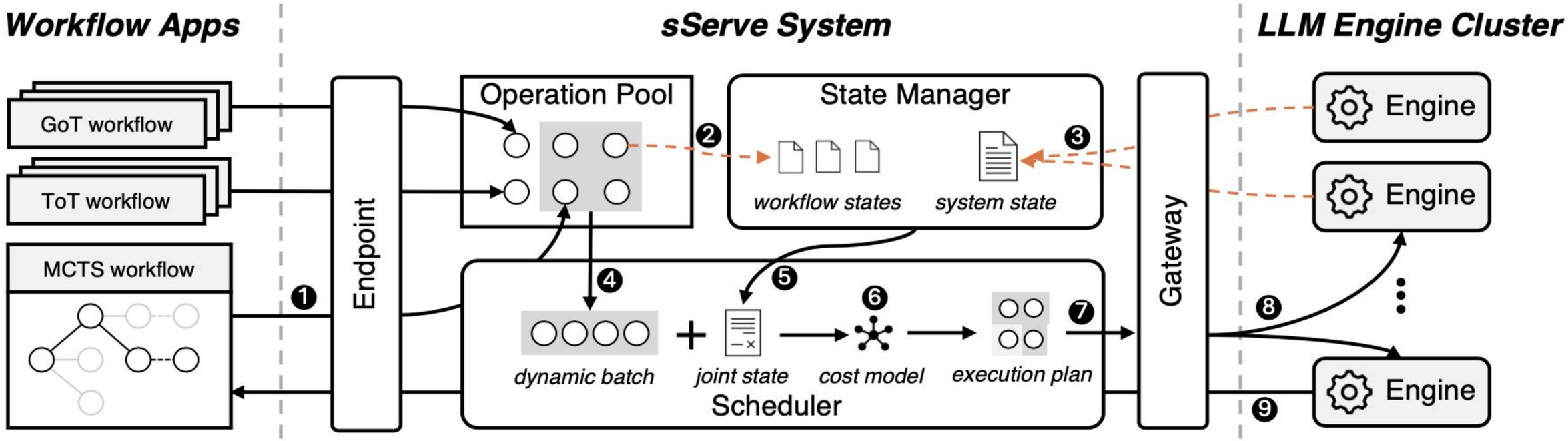
C2 · Adaptive resources

Queues, KV residency and batch compatibility

Joint state guides admission, priority, batching and routing.

sServe: joint state guides workflow scheduling

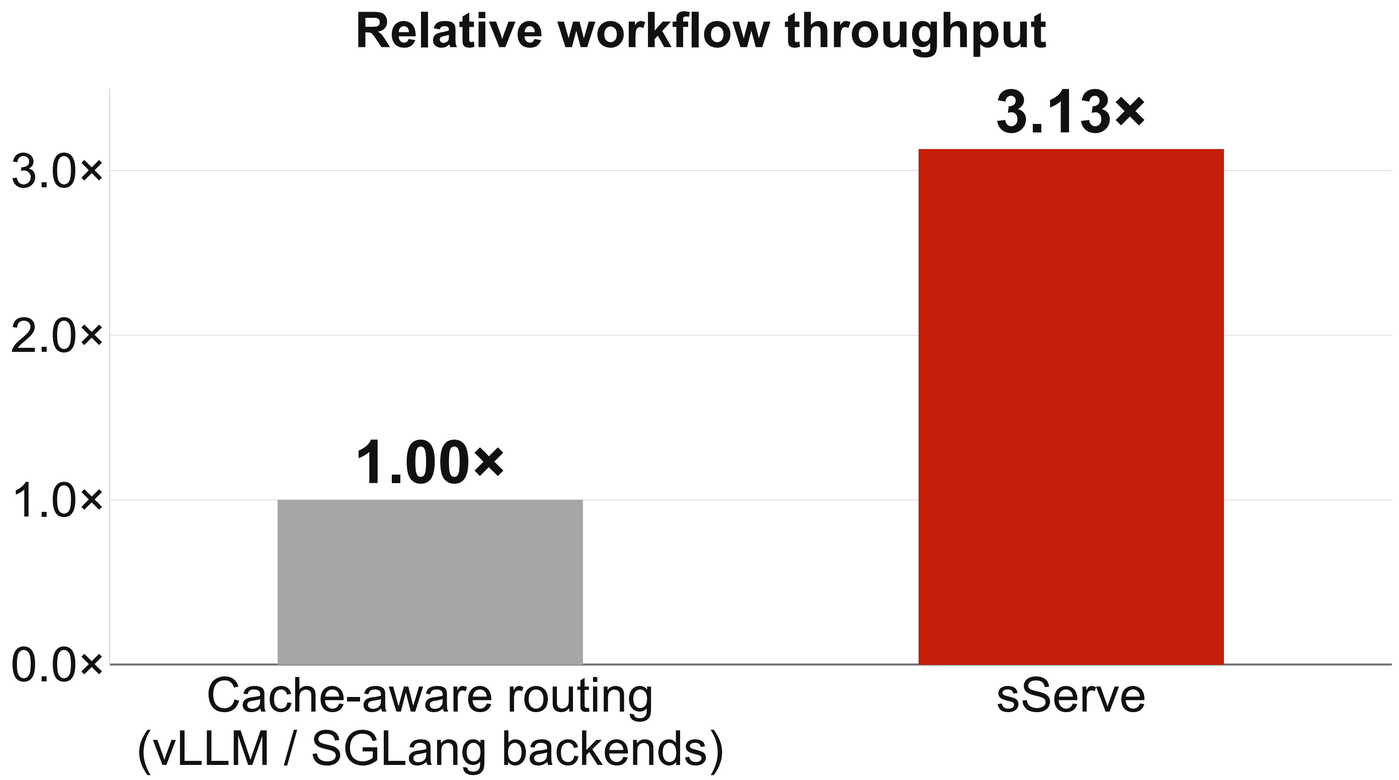
Workflow and backend state guide each scheduling decision



Admission, priority, batching and routing share one state view

sServe: up to 3.13× workflow throughput

Same model, workflow and search budget



Baseline: cache-aware routing (vLLM/SGLang)

Routes requests to reuse cached state.
Does not track workflow dependencies.

Best cason Game-of-24
e: Tree-of-Thoughts

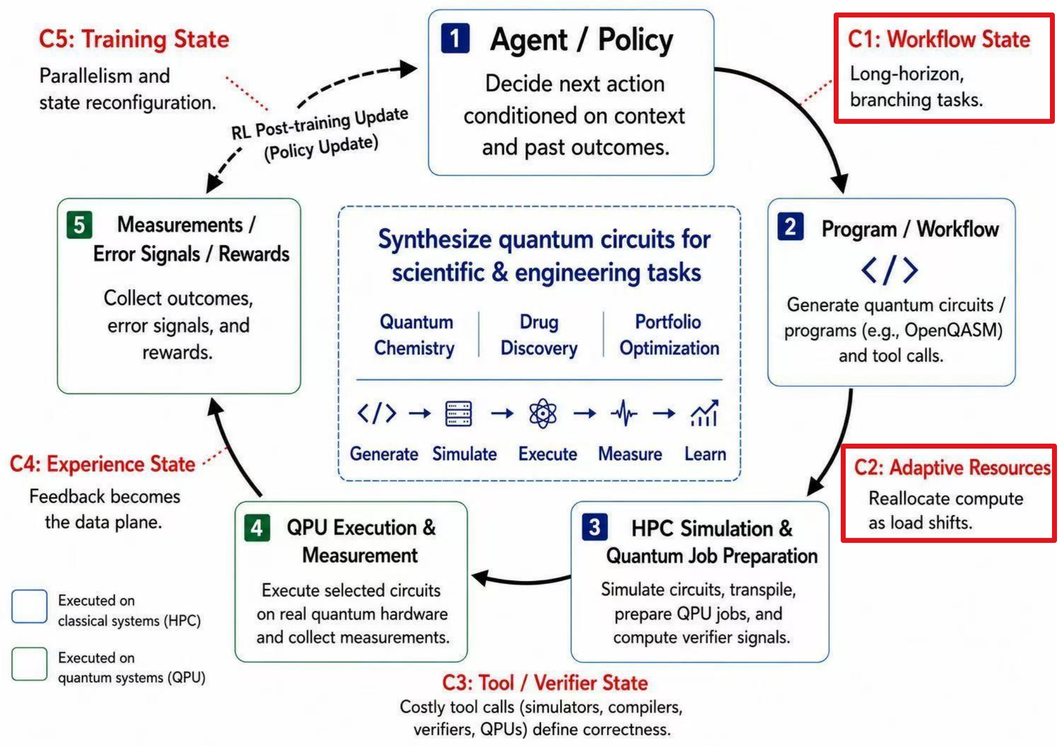
Joint workflow and backend state adds value beyond cache reuse

sServe manuscript · §7.1–7.2, Fig. 3 · Llama-3.1-8B-Instruct · 8 H800 GPUs · Work in progress

ADAPT: EARL follows context growth

EuroMLSys 2026

Location in the research map



C1 · Workflow state

Context grows across interaction turns

↓ **Detect workload drift**

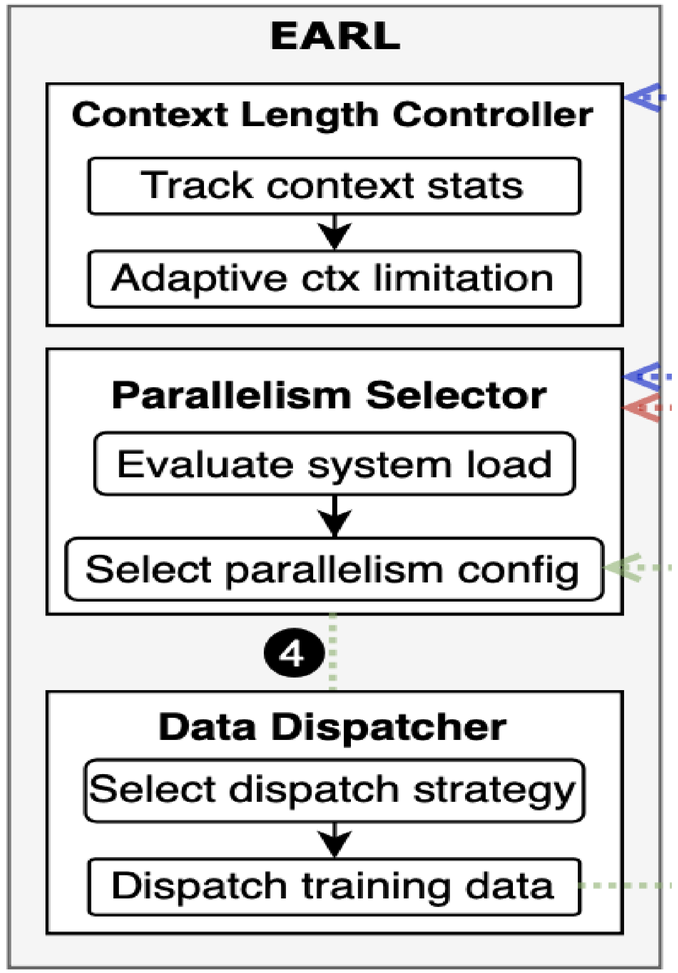
C2 · Adaptive resources

Adapt the runtime configuration

Execution settings should follow the evolving workload.

EARL: how should execution follow growing context?

Context statistics guide parallelism and experience dispatch



Track the changing workload

Track context length and load as interaction turns accumulate.

Choose a suitable GPU layout

Adjust parallelism as memory needs and stage performance change.

Route data to the target shards

Deliver experience directly to the receiving model shards.

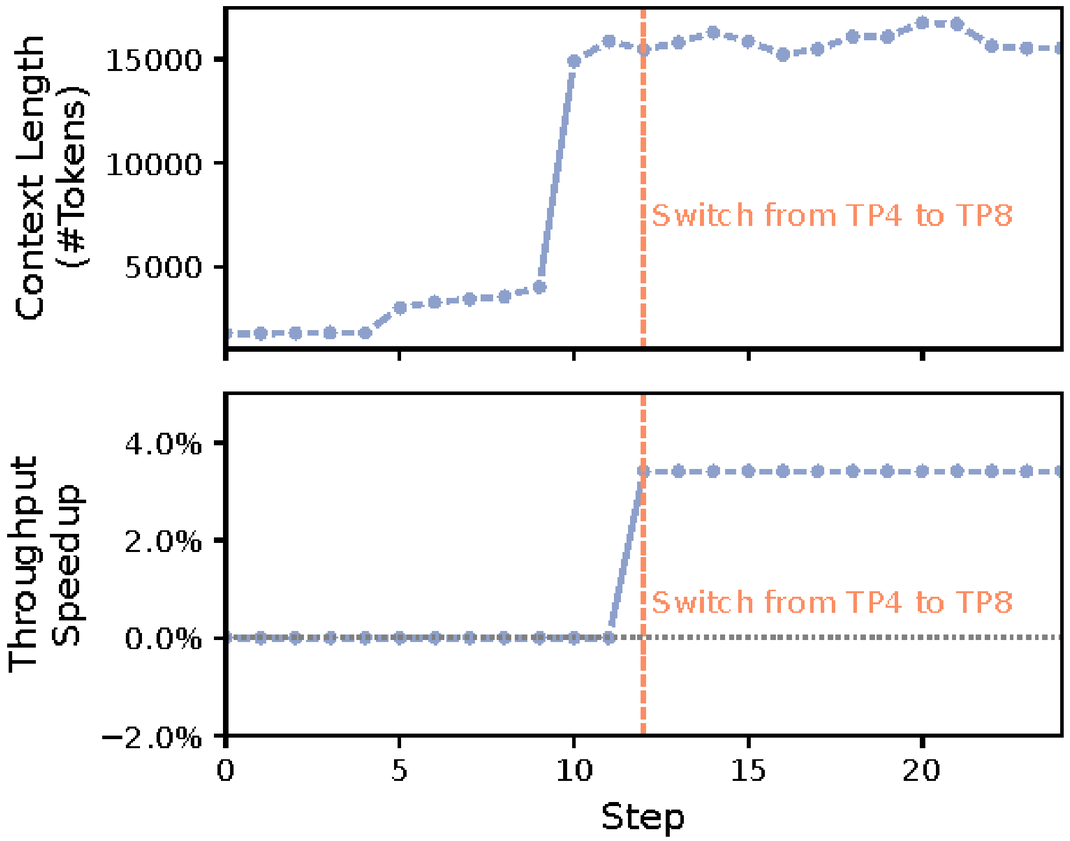
EARL, EuroMLSys 2026. Author-provided controller diagram and Sections 3.1, 3.3–3.4.

EARL: adapt parallelism as context grows

Qwen2.5-72B, Connect-Four, 128 H100 GPUs

Context >16K → TP4 → TP8 at step 12

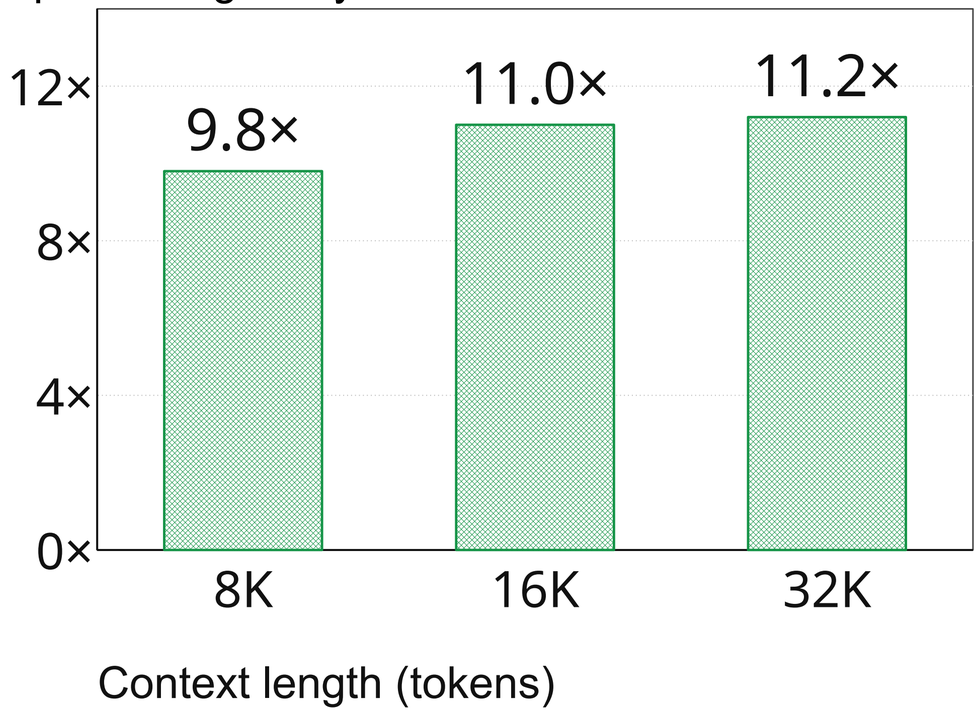
≈3.4% rollout throughput gain vs. fixed TP4



9.8–11.2× faster dispatch

vs. ROLL (Alibaba's LLM RL framework)

dispatch stage only



Direct shard routing

Same TCP/Ethernet transport

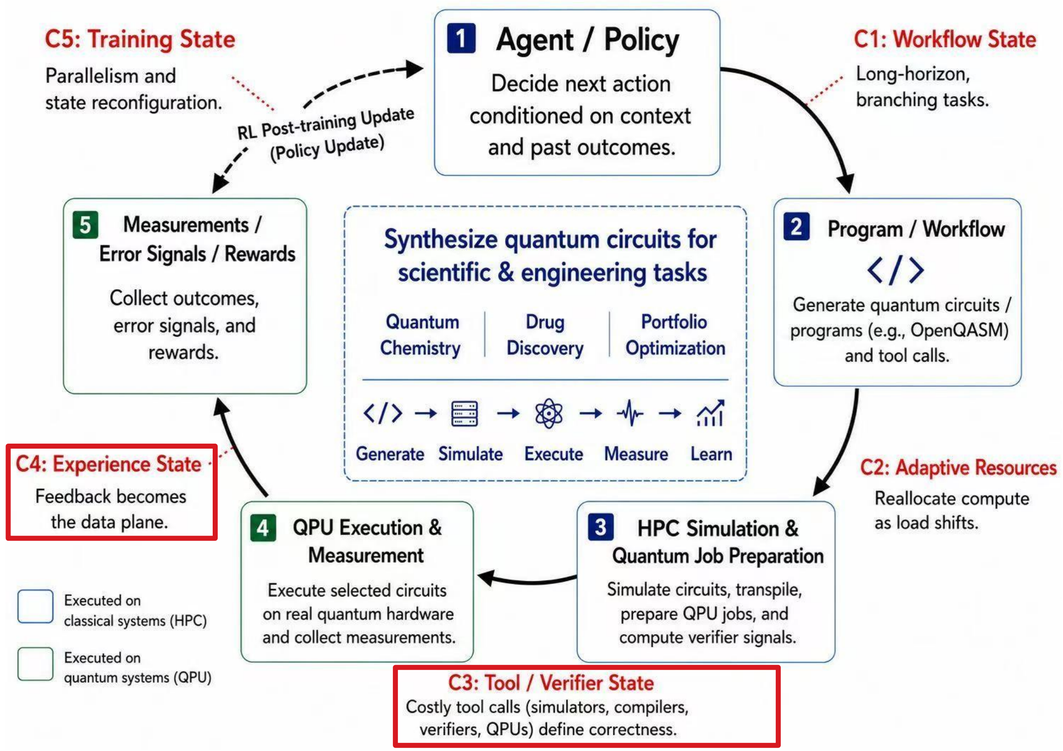
EARL, EuroMLSys '26 · Figures 6–7, Sections 3.1, 3.3–3.4 · doi:10.1145/3805621.3807632

CLOSE: Vista links verification, execution and learning

ACM CAIS 2026

Focus 2 of 2

Location in the research map



C3 · Tool / verifier state

Verification results arrive stage by stage

↓ **Gate costly checks and collect feedback**

C4 · Experience state

Turn verification outcomes into rewards

Decide which check runs next, then use feedback to improve learning.



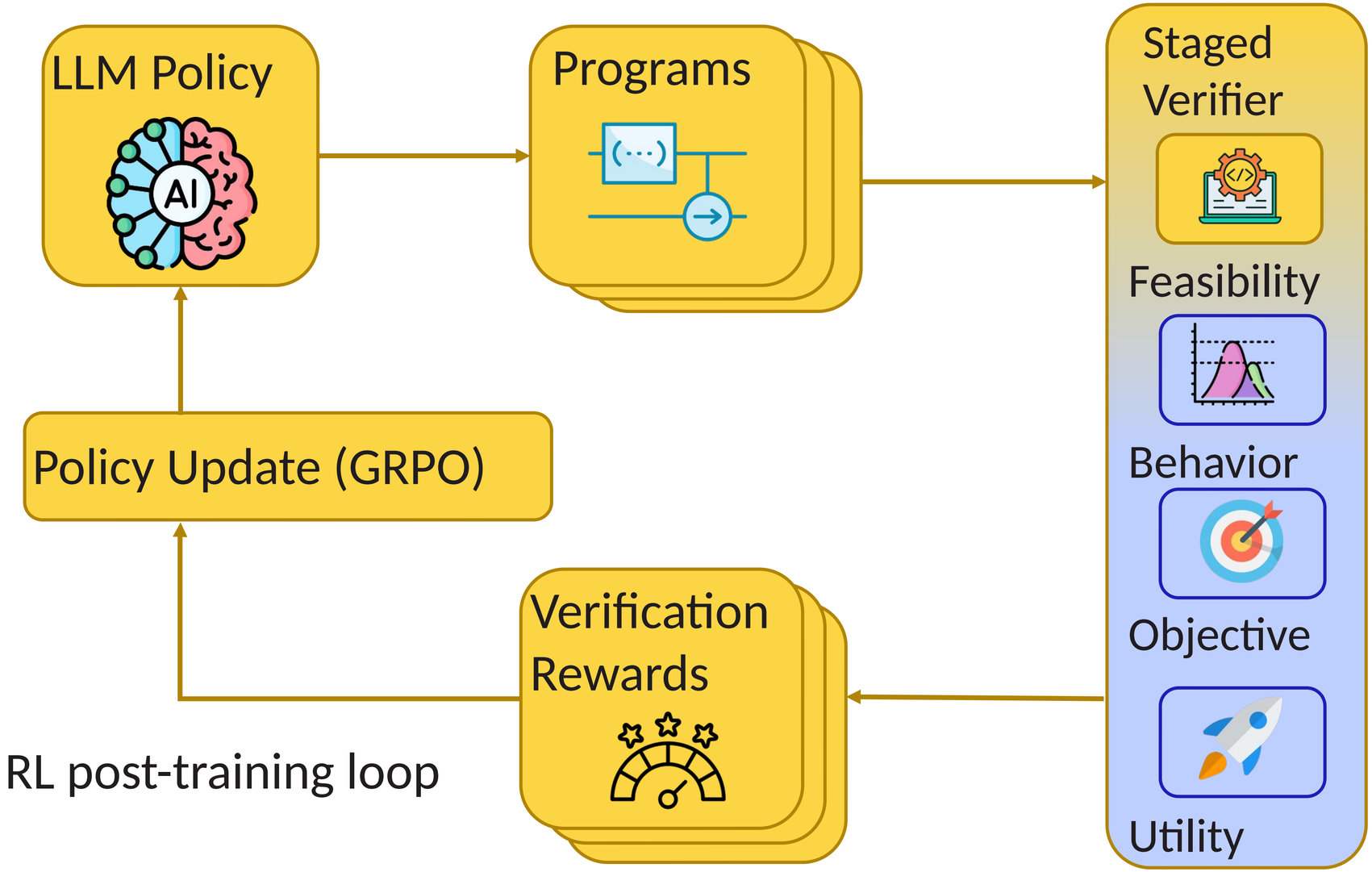




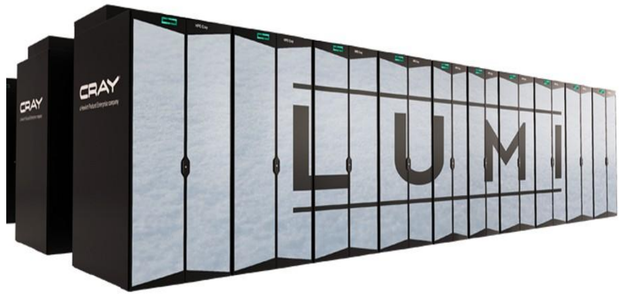
ACM Conference on AI and Agentic Systems

Vista: verification results become learning signals

Verification decides the next check and supplies learning signals



Classical



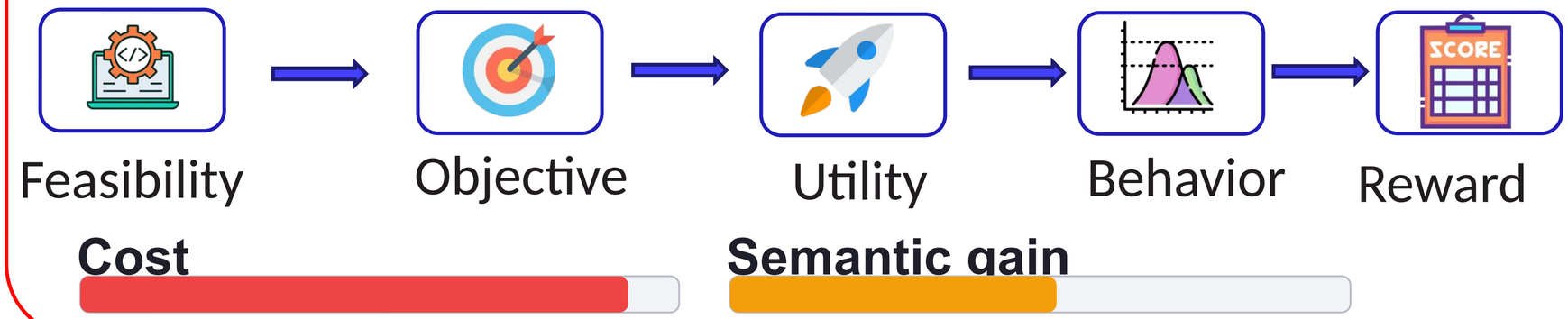
Quantum





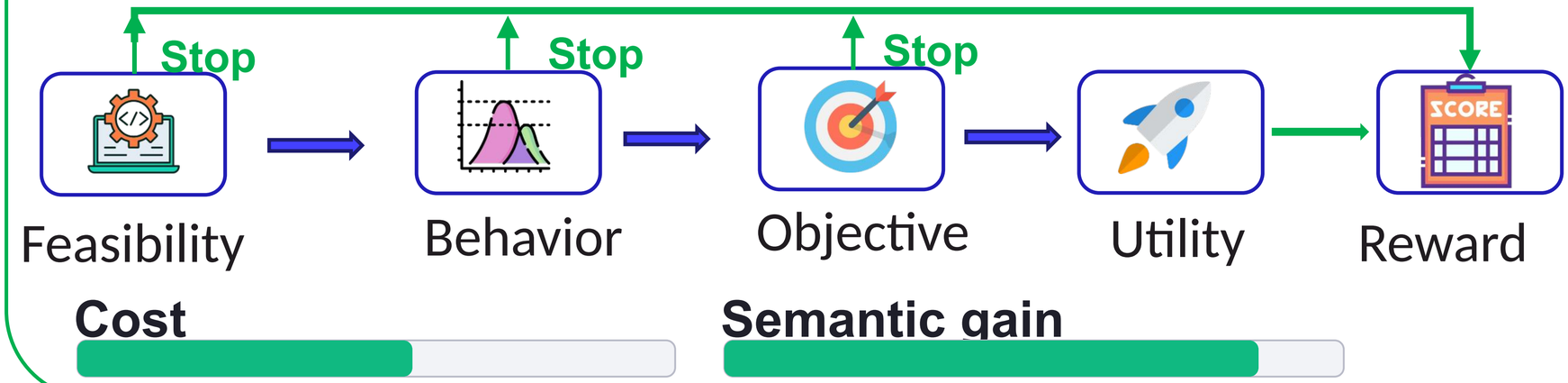
Vista: which candidate deserves the next check?

Budget-unaware



Late screening wastes costly checks.

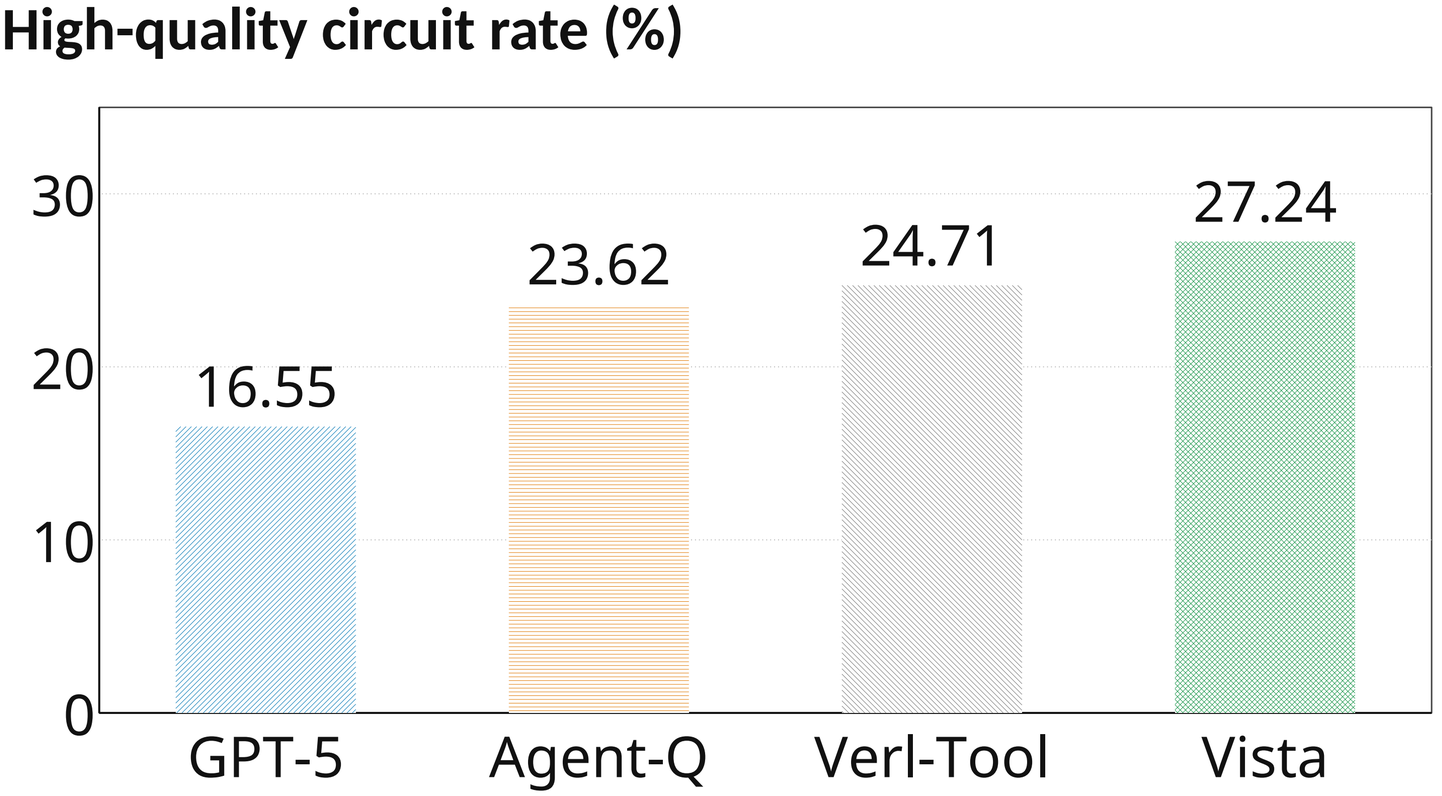
Budget-aware



Early evidence guides the next check.

Vista: verification improves circuit quality

High-quality circuit rate at Pass@10, higher is better



100% executability at Pass@10

33.10% objective success

27.24% high-quality circuit rate

Verification improves both executability and semantic quality

Vista, ACM CAIS '26 · Table 1 · doi:10.1145/3786335.3813148

Vista: 1.63× end-to-end speedup on LUMI + Q50



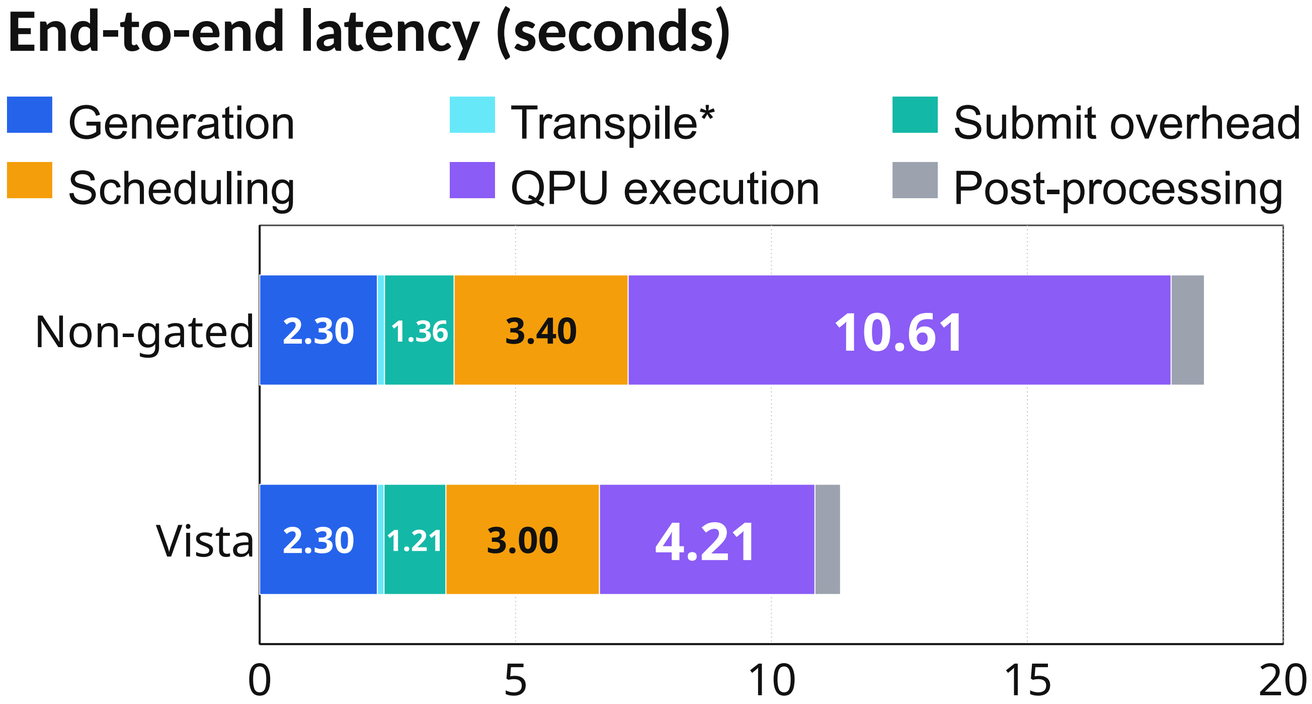
VTT Q50

Source: https://lumi-supercomputer.eu/vttq50-connected-to-lumi/



LUMI

Source: https://lumi-supercomputer.eu/deep-dive-into-the-building-of-the-lumi-data-center/



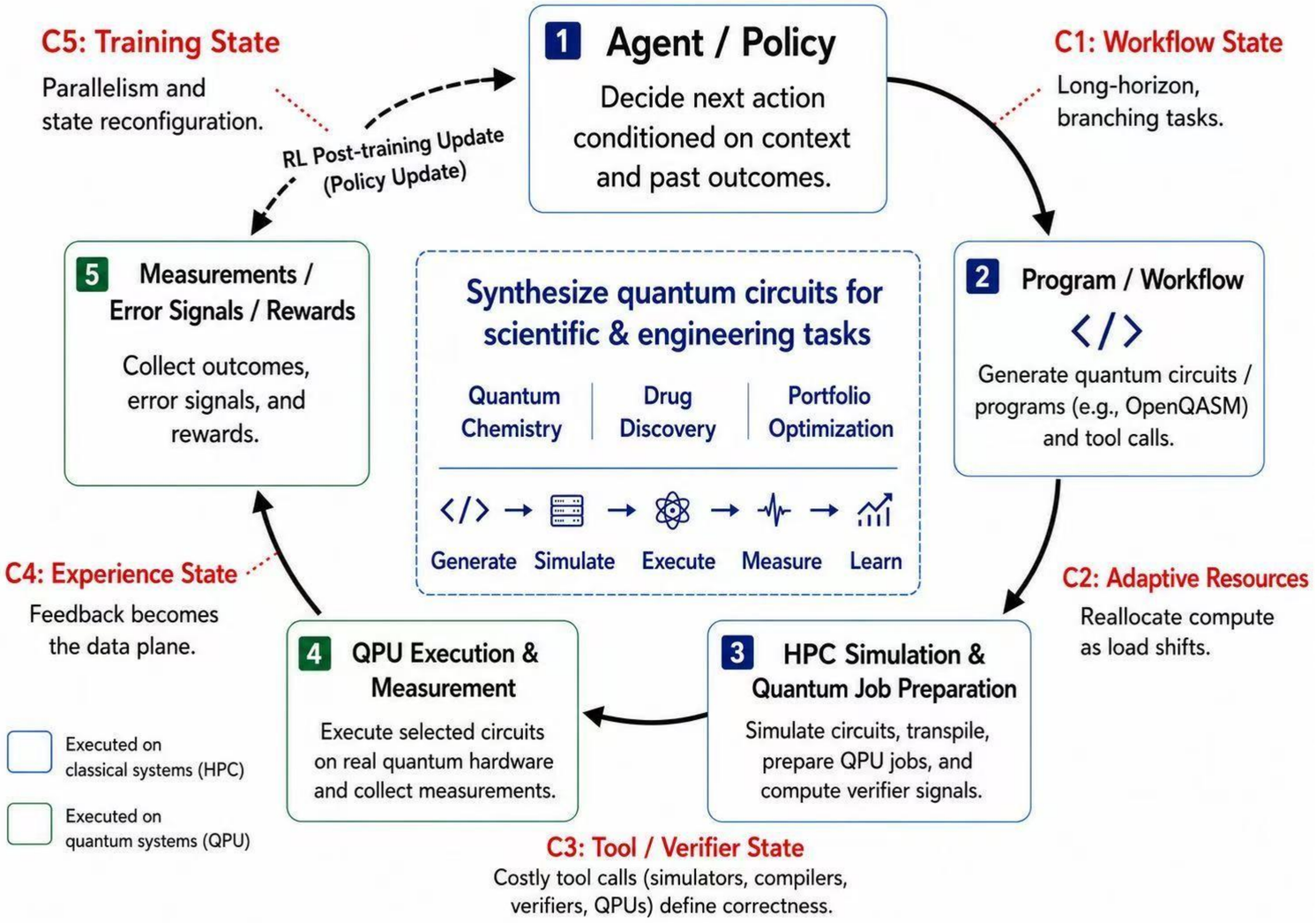
*Transpile: residual from reported totals.

1.63× end-to-end speedup
18.46 → 11.35 s on LUMI + Q50

Selective verification reduces QPU time

Conclusion: State should guide the next runtime decision

State Management Across the Agentic AI Lifecycle



C2 + C5

Adapt computation
MSRL, Tenplex, Nereus

C1 + C2 + C4

Schedule data and workflows
Conduit, sServe, EARL

C3 + C4 + C5

Turn verification into feedback
Vista