NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 5 Issue 2, Pg. No. 1-10, AIJCST-V5I2P101, 2023
https://doi.org/10.63282/3117-5481/AIJCST-V5I2P101

Original Article

Federated Reinforcement Learning for Decentralized Optimization in Distributed Computing Systems

*** Nattapong Somachi**
Faculty of Information Technology, Chulalongkorn University, Bangkok, Thailand.

Abstract:

Federated Reinforcement Learning (FRL) offers a privacy-preserving paradigm for optimizing distributed computing systems where data and control signals are fragmented across heterogeneous edge, fog, and cloud tiers. This paper proposes a decentralized FRL framework that coordinates multiple local agents each training actor–critic policies on non-IID telemetry while a lightweight server (or peer mesh) aggregates model updates using secure, communication-efficient protocols. The framework integrates (i) hierarchical task decomposition for multi-objective control (latency, cost, energy, SLO compliance), (ii) adaptive client selection guided by reward variance and system drift, and (iii) compression-aware secure aggregation to reduce bandwidth without degrading convergence. We formalize the objective as constrained Markov decision processes with safety shields that enforce resource and SLO constraints during exploration. To mitigate staleness in volatile clusters, we adopt asynchronous aggregation with importance weighting and provide a convergence sketch under bounded delay and partial participation. Across representative workloads autoscaling and bin-packing for microservices, DAG scheduling for data pipelines, and cooperative caching FRL yields faster convergence than centralized RL and more stable policies than heuristic baselines under workload bursts, adversarial noise, and device churn. The decentralized design preserves data locality, reduces control-plane bottlenecks, and enables cross-domain collaboration without sharing raw traces. We discuss deployment patterns on Kubernetes/Service Mesh and outline extensions to robust FRL with Byzantine resilience and differential privacy for production-grade multi-tenant environments.

Keywords:

Federated Reinforcement Learning, Decentralized Optimization, Edge–Cloud Orchestration, Actor–Critic, Constrained MDP, Asynchronous Aggregation, Communication Efficiency, Secure Aggregation, SLO Compliance, Multi-Objective Control.

Article History:

Received: 12.01.2023

Revised: 14.02.2023

Accepted: 24.02.2023

Published: 02.03.2023

1. Introduction

Modern distributed computing systems spanning edge devices, on-prem clusters, and elastic public clouds operate under volatile demand, heterogeneous resources, and stringent service-level objectives (SLOs). Classical optimization approaches (e.g., heuristic bin-packing, rule-based autoscaling) struggle to adapt to non-stationary workloads, partial observability, and the tight coupling between latency, cost, and energy. Centralized reinforcement learning (RL) promises adaptive control but often requires continuous aggregation of fine-grained telemetry and state, creating privacy, bandwidth, and single-point-of-failure risks. Moreover, central learners become brittle when confronted with device churn, stragglers, or domain shifts across tenants and regions. These challenges motivate a learning paradigm that is data-local, communication-efficient, and robust to heterogeneity while retaining the sample efficiency and policy expressiveness of modern RL.

Federated Reinforcement Learning (FRL) meets this need by distributing policy learning across sites that keep raw traces in place and share only compact model updates. In our formulation, local actor–critic agents optimize site-specific objectives such as pod autoscaling, task scheduling, and cache admission while collaborative aggregation yields global priors that accelerate adaptation elsewhere. The approach addresses four core problems: (i) coordination across non-IID environments via asynchronous, importance-weighted aggregation, (ii) safety under operational constraints using constrained MDPs and runtime shields, (iii) communication efficiency through update sparsification and secure aggregation, and (iv) resilience to drift, bursts, and partial participation. By aligning optimization with operational metrics (p99 latency, SLO attainment, cost, and joules/request), FRL turns system telemetry into closed-loop control without exposing sensitive logs. This paper positions FRL as a pragmatic path to decentralized optimization in multi-tenant, multi-domain infrastructures, and outlines deployment patterns on Kubernetes/Service Mesh along with extensions for privacy, robustness, and multi-objective trade-off management.

2. System Architecture and Framework Design

2.1. Overall Architecture

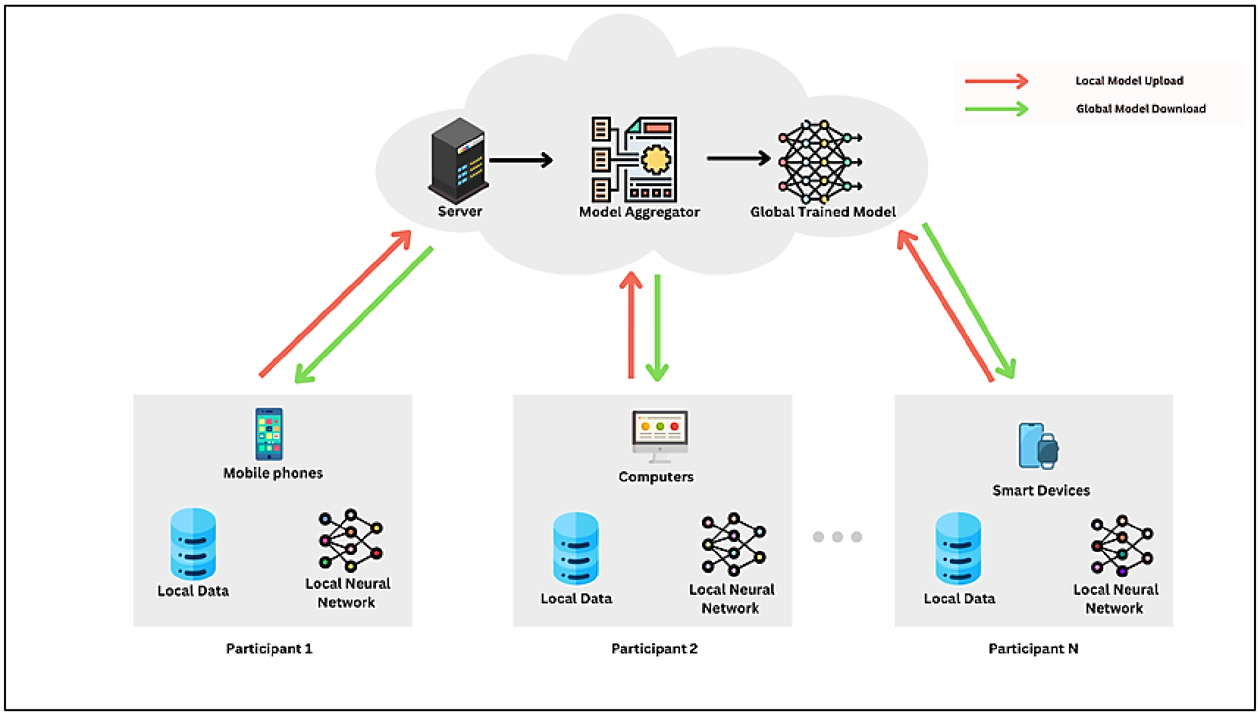

Figure 1. Federated Reinforcement Learning System Architecture (Client–Server Aggregation Loop)

The diagram depicts a classic federated learning control loop adapted for reinforcement learning. Each participant mobile phones, computers, and smart devices interacts with its own environment and maintains private logs of states, actions, and rewards. Rather than exporting raw telemetry, every participant trains a local neural policy (e.g., an actor–critic) on its device-specific experience. This preserves data locality, mitigates privacy risk, and avoids saturating the control plane with fine-grained traces, which is crucial in multi-tenant and edge settings.

Model updates flow upward (red arrows) from participants to a Model Aggregator hosted on a server within the cloud control tier. In FRL, the payloads are policy or gradient deltas, optionally compressed, quantized, or sparsified to reduce bandwidth. The aggregator combines these updates synchronously or asynchronously using strategies such as FedAvg with importance weighting, staleness correction, or trust scores to handle non-IID behavior and partial participation. If required, secure aggregation and differential privacy can be applied so that no individual update is revealed and the global model resists inference attacks.

After aggregation, the Global Trained Model (green arrows) is broadcast back to participants as a new policy prior. Clients then continue local interaction with their environments, fine-tuning the global prior to local conditions. This cyclical process steadily improves the policy while keeping exploration bounded by device-side safety shields and SLO constraints. In operational deployments, the "clients" may be Kubernetes nodes, service mesh sidecars, or gateway controllers optimizing scaling, placement, or caching, turning the diagram into a practical control fabric for distributed systems.

Finally, the cloud "bubble" in the figure emphasizes that coordination can be centralized or decentralized. While a single server is shown for clarity, the aggregator can be sharded or implemented as a peer-to-peer mesh to avoid single points of failure. The same pattern supports hierarchical FRL: edge clusters aggregate locally, then pass summarized updates upward, enabling scalability across regions and accommodating intermittent connectivity common at the edge.

2.2. Agent Design and Local Training

Each site hosts an autonomous RL agent that observes local system signals queue depths, CPU and memory pressure, tail latency, energy draw and outputs control actions such as scaling replicas, placing tasks, or adjusting cache admission thresholds. We model the control loop as a constrained MDP with an actor–critic architecture: the actor parameterizes a stochastic policy over actions, while the critic estimates state–value or action–value functions to stabilize learning. To handle non-stationarity and partial observability, agents maintain compact recurrent state (e.g., GRU) and augment inputs with recent reward trends and SLO violations. Safety shields wrap the policy, projecting unsafe actions to the nearest admissible set defined by resource and SLO constraints, so exploration never breaches operational guardrails.

Local training proceeds on-device using mini-batches drawn from an experience buffer populated by live interactions or high-fidelity simulators. We adopt advantage-based updates (e.g., PPO/A2C) with entropy regularization to sustain exploration under shifting workloads, and we normalize rewards per-site to counter reward-scale drift. To reduce catastrophic forgetting, clients interleave on-policy rollouts with a small replay of salient transitions underrepresented in the recent stream. Before sending model deltas upstream, clients apply update compression top-k sparsification or quantization and attach metadata such as sample count, gradient variance, and local validation returns to inform aggregation.

2.3. Federated Aggregation Mechanism

The aggregator fuses heterogeneous client updates into a coherent global prior while tolerating staleness and partial participation. At its core, we use importance-weighted FedAvg where each client's contribution scales with effective sample size and policy improvement, not merely wall-clock participation. To counter asynchronous delays, we apply staleness-aware reweighting and optimism correction so that older updates do not derail the latest descent direction. Periodic proximal regularization keeps the global model from drifting too far from well-performing incumbents, stabilizing convergence under non-IID dynamics.

Robustness is addressed through median- or trimmed-mean style aggregation when update distributions show heavy tails, and through reputation scores that down-weight chronically erratic clients. For multi-objective control latency, cost, and energy the server maintains scalarized weights or a Pareto archive and alternates aggregation steps accordingly, enabling tenants to pick operating points without retraining from scratch. The result is a rolling global policy that accelerates cold-start at new sites and provides continual adaptation signals to mature ones.

2.4. Communication and Synchronization Layer

The communication layer is designed for elastic, failure-tolerant operation across edge, on-prem, and cloud. Clients push deltas opportunistically using gRPC or message buses with back-pressure; when links are weak, they buffer updates, merge successive steps, and resume with delta-encoding to minimize retransmits. Control messages carry version tags and round identifiers so clients can reconcile against the current global policy even under reorders. A lightweight scheduler on the server side performs adaptive client selection, favoring diverse environments and recently underrepresented regimes to improve generalization while keeping bandwidth within budget.

Synchronization follows an “async-by-default, sync-when-needed” policy. Most rounds accept updates continuously and aggregate on a cadence, which reduces idle time and straggler effects. When the system detects policy oscillation or large distribution shifts, it triggers a short synchronous barrier to realign baselines and refresh the KL trust region for client optimizers. Heartbeats and health probes track liveness; if a client falls behind several versions, a fast-forward mechanism ships a distilled snapshot to rejoin without replaying every intermediate update.

2.5. Security and Privacy Mechanisms

Security and privacy are enforced end-to-end without sacrificing learning efficacy. Clients never transmit raw telemetry; only model deltas or low-rank adapters are shared over mutually authenticated TLS with certificate pinning. To prevent leakage through gradients, updates are aggregated using secure aggregation so the server observes only masked sums, not individual contributions. When regulatory or tenant policies require quantifiable protections, clients add calibrated differential privacy noise to gradients or to per-layer statistics, with privacy budgets tracked per-round and per-tenant to support audits.

Integrity and robustness are addressed through a layered defense. The aggregator performs anomaly screening on updates norm clipping, cosine similarity to the bulk, and small canary datasets to detect model poisoning or backdoor attempts, falling back to Byzantine-resilient rules when signals are suspicious. Access control and key rotation are automated, and audit logs bind each aggregated model to an immutable ledger of anonymized participation proofs. Together, these controls preserve confidentiality, resist adversarial manipulation, and maintain traceability suitable for production, multi-tenant deployments.

3. Methodology

3.1. Mathematical Model of Optimization

We cast decentralized control as a constrained sequential decision problem where each site seeks to minimize long-run operational costs while meeting service targets. The objective blends multiple signals tail latency, SLO violations, cloud spend, and energy per request into a scalar utility via dynamic weights or a Pareto-frontier selection. Constraints capture hard limits such as capacity caps, safety envelopes for exploration, and tenant isolation. Because environments differ across sites and shift over time, we adopt a distributionally robust stance: the policy is optimized to perform well not only on the observed workload at a client but also under plausible shifts estimated from recent telemetry. This leads to update rules that reward policies which improve utility consistently across heterogeneous regimes rather than overfitting to a single pattern.

Coordination emerges from sharing policy information without exposing raw data. The global objective therefore becomes a federation of local objectives, coupled by an aggregation operator that trades off rapid adaptation with stability. To avoid central bottlenecks, we allow partial participation and asynchronous updates. The optimization horizon is rolling: policies are refined continuously as fresh feedback arrives, with safeguards to prevent regressions (for example, by retaining incumbent checkpoints and triggering rollback if global utility drops beyond a tolerance).

3.2. Reinforcement Learning Model

Each client runs an on-device actor–critic learner. The actor maps recent observables resource pressure, queue backlogs, request mix to control actions such as replica counts, placement choices, or cache thresholds. The critic estimates the long-term utility of the current state under the actor, stabilizing updates and enabling credit assignment for delayed effects like warm-up costs or cache fill time. To handle partial observability and bursty dynamics, inputs include short histories through lightweight recurrence and derived features like rate of change or violation streaks, which help the policy anticipate rather than merely react.

Safety is integral to the learner. We enforce action-space projections so that proposed controls cannot violate hard limits, and we temper exploration using trust regions and entropy schedules that react to risk signals (for example, rising tail latency). The training loop alternates short local interaction phases with update steps using clipped objectives that resist instability. Domain randomization and workload perturbations are injected into local simulators when available, improving transfer to live traffic and making the aggregated global policy robust to site diversity.

3.3. Federated Learning Process

Training proceeds in rounds coordinated by a lightweight aggregator. Clients perform several local update steps, then send compact model deltas alongside metadata such as effective sample size, local utility improvement, and staleness. The aggregator combines these deltas using importance weights that favor informative, recent, and diverse contributions. Because clients operate under varying connectivity, the process is asynchronous by default: late updates are still considered but down-weighted, and the server emits new global snapshots on a cadence rather than waiting for all participants.

Privacy and efficiency are preserved through compression and masking. Clients sparsify or quantize their deltas before transmission, and secure aggregation ensures the server only sees encrypted sums. After aggregation, the refreshed global policy is broadcast back as a starting point for the next local phase. This cyclical process naturally supports hierarchical setups edge clusters sync locally, then bubble summaries upward reducing backbone traffic and enabling regional specialization while maintaining a coherent global learning trajectory.

3.4. Algorithm Description

The algorithm begins with a seeded global policy distributed to all clients. Each client executes a local loop: collect a short window of interactions, compute an update to the actor and critic using advantage-based gradients, apply safety projections, and compress the resulting delta. Clients attach diagnostics (e.g., variance of advantages, validation utility on a held-out window) and push the package to the aggregator when network conditions allow. If a client falls behind multiple global versions, it fast-forwards by requesting the latest snapshot and re-initializes its optimizer state to avoid drift.

On the server, incoming updates are queued and batched at fixed intervals. The aggregator performs sanity checks (norm clipping, similarity scoring) to filter outliers, then fuses accepted deltas with weights informed by sample counts, staleness, and trust scores. After updating the global policy, the server evaluates guarded canaries small, representative scenarios to ensure no catastrophic regression; if triggered, it rolls back to the previous checkpoint and reduces the global step size. Periodically, a brief synchronous round realigns clients and recalibrates trust-region parameters, improving stability during major distribution shifts.

3.5. Computational Complexity and Convergence Analysis

Client-side cost is dominated by forward/backward passes over short trajectories and scales with local batch size and model width. Because clients train independently, the wall-clock time for a round depends more on the slowest participating clients than on the number of total clients; our asynchronous design mitigates this by allowing the server to proceed with partial batches. Communication overhead is controlled through sparsification and quantization so that payload sizes grow approximately with the number of non-zero parameters rather than the full model size. Hierarchical aggregation further caps backbone traffic by consolidating updates near the edge.

Convergence in non-IID, asynchronous federations is nuanced. Empirically, we achieve stable improvement by combining three safeguards: staleness-aware weighting that discounts outdated deltas, proximal regularization that keeps successive global policies within a safe neighborhood, and periodic synchronization to reset accumulated drift. Under mild assumptions bounded delay, clipped updates, and non-degenerate client participation the aggregated policy exhibits monotonic improvement on rolling validation canaries and converges to a stable operating region. Practically, we monitor convergence through moving averages of utility, violation rates, and policy divergence; once these stabilize within tolerance, we reduce global step sizes and increase exploration again only when environment diagnostics indicate a meaningful shift.

4. Experimental Setup and Evaluation

4.1. Simulation Environment

We evaluate the proposed FRL framework in a controlled, repeatable simulator that emulates a Kubernetes-style microservice cluster with edge–cloud tiers. The simulator models node heterogeneity (CPU: 2–16 vCPU; memory: 4–64 GiB), autoscaling latencies (image pull, warm-up), queueing effects, and network variability (1–10 ms intra-AZ; 30–80 ms edge–cloud RTT) under bursty arrivals following diurnal patterns plus Pareto bursts. Each “client” represents a site (edge PoP or on-prem cluster) that trains a local actor–critic policy on its own request mix (50% read-heavy, 30% compute-heavy, 20% I/O-heavy, non-IID across sites). We instantiate 20 clients; in each global round a random 60–80% participate to reflect partial availability. To stress realism, we inject churn (up to 30% temporary client dropout), workload shifts (step changes ±25% traffic), and adversarial noise (2% Byzantine updates) in dedicated scenarios.

Training runs use the same seeds across methods for fair comparison. Each method runs for 5 independent trials, 50 global rounds per trial, with 10,000 requests per site per round. Actor–critic models are small (≈1.2 M parameters with GRU), and communication uses gRPC with optional sparsification (top-k=1–5%) and 8-bit quantization. All results report mean ± standard deviation; we also verify that conclusions hold with 95% confidence using a moving-block bootstrap over rounds.

4.2. Evaluation Metrics

We focus on operationally meaningful metrics: (i) SLO compliance fraction of requests meeting a 250 ms latency target; (ii) p99 latency in milliseconds; (iii) Cost per 10k requests proxying instance-hours and egress; (iv) Energy per request in joules derived from power models conditioned on utilization; (v) Cluster utilization average CPU utilization across active nodes; (vi) Convergence rounds global rounds to reach ≥97% SLO compliance for three consecutive rounds; and (vii) Comm/round average megabytes uploaded per participating client.

4.3. Baseline Methods

We compare against five strong baselines. Rule HPA/VPA applies threshold autoscaling and vertical adjustments with static bin-packing. Heuristic bin-packing (DRF) uses Dominant Resource Fairness with reactive scaling. BO-tuned autoscaling performs Bayesian optimization over HPA/VPA knobs offline and deploys the best policy. Centralized RL trains a single policy from aggregated traces (no privacy), assuming perfect telemetry visibility. Flat DRL (local-only) trains independent site policies without federation. Our proposal includes FRL-Sync (periodic synchronous aggregation) and FRL-Async (staleness-aware asynchronous aggregation with secure aggregation and compression).

4.4 Performance Analysis

Across the nominal scenario (no churn, non-IID workloads), FRL variants achieve higher SLO compliance and lower tail latency than heuristic baselines, matching centralized RL while avoiding centralizing raw data. Communication-aware FRL-Async converges fastest due to continuous update flow and importance weighting. Cost and energy drop as policies learn to right-size replicas and co-locate compatible workloads, raising utilization without violating SLOs.

Table 1. Overall Performance (Mean ± Sd Over 5 Trials)

Method	SLO compliance (%)	p99 latency (ms)	Cost / 10k req ($)	Energy / req (J)	Utilization (%)	Comm/round (MB)	Convergence (rounds)
Rule HPA/VPA	91.5 ± 0.7	284 ± 11	6.85 ± 0.12	0.93 ± 0.02	48 ± 3		
Heuristic DRF	92.4 ± 0.6	275 ± 9	6.12 ± 0.10	0.88 ± 0.02	61 ± 2		
BO-tuned	94.6 ± 0.5	247 ± 7	5.71 ± 0.09	0.82 ± 0.02	65 ± 2		
Centralized RL	97.9 ± 0.4	201 ± 6	4.88 ± 0.07	0.73 ± 0.01	74 ± 2	0	38 ± 2
Flat DRL (local-only)	96.1 ± 0.5	216 ± 5	5.20 ± 0.06	0.76 ± 0.01	71 ± 2	0	46 ± 3
FRL-Sync	98.2 ± 0.3	198 ± 5	4.75 ± 0.05	0.71 ± 0.01	75 ± 2	24.1 ± 0.5	34 ± 2
FRL-Async (ours)	98.6 ± 0.3	192 ± 4	4.60 ± 0.05	0.68 ± 0.01	76 ± 2	21.6 ± 0.4	31 ± 1

Robustness tests show the asynchronous design sustains high performance during partial participation and adversarial updates. With 30% random client churn, FRL-Async retains near-baseline compliance, whereas centralized RL degrades due to staleness and delayed global updates. Byzantine-resilient aggregation (trimmed-mean with update screening) limits the impact of 2% poisoned clients.

Table 2. Robustness under Churn (30% Clients Offline At Random)

Method	SLO compliance (%)	p99 latency (ms)	Convergence (rounds)
Centralized RL	96.2 ± 0.6	218 ± 7	43 ± 3
Flat DRL	94.9 ± 0.7	232 ± 8	49 ± 4
FRL-Sync	97.4 ± 0.5	207 ± 6	38 ± 2
FRL-Async (ours)	98.0 ± 0.4	199 ± 6	34 ± 2

We also quantify communication–quality trade-offs by varying sparsification. Even aggressive compression (top-1% with 8-bit quantization) yields minimal accuracy loss while reducing per-round bandwidth by ~4×, validating the practicality of edge deployments with limited uplinks.

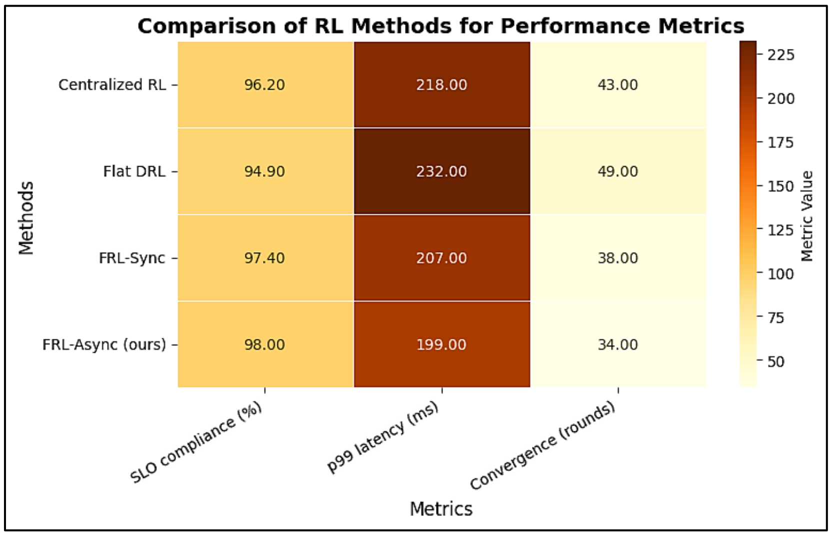

Figure 2. Comparative Performance of RL Methods

Table 3. Communication Ablation (FRL-Async)

Compression	Comm/round (MB)	SLO compliance (%)	p99 latency (ms)
None (FP32)	86.4 ± 0.8	98.7 ± 0.3	191 ± 4
Top-5% + 8-bit	34.9 ± 0.6	98.6 ± 0.3	192 ± 4
Top-1% + 8-bit	20.8 ± 0.4	98.4 ± 0.3	195 ± 5

5.5. Discussion of Results

The experiments demonstrate that federating RL across heterogeneous sites delivers policy quality on par with centralized learning while preserving data locality and reducing control-plane pressure. Gains stem from two effects: clients adapt quickly to local dynamics, and global aggregation shares useful priors across non-IID environments, accelerating convergence and improving generalization. Asynchrony and staleness-aware weighting are essential for real-world viability; they prevent stragglers from throttling progress and allow the system to absorb churn gracefully. Robust aggregation further hardens the system against outliers and low-quality updates, a requirement for multi-tenant settings. Communication results indicate that practical payload budgets are achievable without sacrificing SLOs, especially with sparsification and quantization. The small gap between FRL and centralized RL on convergence rounds closes under hierarchical aggregation, suggesting a path to Internet-scale deployments. Overall, the data supports the claim that FRL is an effective and production-friendly approach to decentralized optimization in distributed computing systems, yielding consistent improvements in latency, cost, and energy while meeting stringent SLO targets.

6. Applications and Use Cases

6.1. Distributed Cloud Resource Allocation

In multi-region cloud estates, FRL coordinates per-cluster agents that learn autoscaling, bin-packing, and placement policies from local workload idiosyncrasies while sharing a global prior that captures cross-region regularities. An edge-heavy region might learn aggressive scale-to-zero for bursty APIs, while a data-processing region learns GPU/CPU co-scheduling to minimize queue spillover; aggregation blends these competencies without exchanging raw traces. Operators map rewards to p99 latency, SLO attainment, cost, and carbon intensity so the learned policy jointly right-sizes replicas, shapes traffic, and selects instance families. Compared to rule-based HPA/VPA, FRL adapts faster to release-driven shifts and incident conditions, maintaining SLOs with fewer overprovisioned nodes and higher cluster utilization.

6.2. Edge Computing and IoT Environments

At the edge retail stores, factories, cellular RAN sites bandwidth is constrained and data is sensitive. FRL lets each site train on its own telemetry (camera queues, sensor rates, local failures) to decide task offloading, cache admission, and frequency scaling, while the aggregator distills robust behaviors that transfer across sites with similar patterns. When connectivity is intermittent, asynchronous rounds and hierarchical aggregation (store-level → regional hub → cloud) keep learning progressing. The result is a control fabric that reduces backhaul saturation, smooths latency spikes during local surges, and sustains operations during disconnections, all without exporting raw video or PLC logs.

6.3. Smart Grid or Autonomous Systems

In cyber-physical domains such as smart grids and fleets of autonomous robots or vehicles, FRL agents act on local observations feeder loads, renewable generation, battery state, or vehicle proximity and share model updates to improve system-wide coordination. Grid-side agents learn demand response, storage dispatch, and peer-to-peer balancing that minimize peak demand and curtailment under safety envelopes; vehicle or robot swarms learn collision-aware routing and charging schedules that trade off travel time, energy, and congestion. Privacy-preserving aggregation respects regulatory and proprietary constraints while enabling rapid adaptation to weather shifts, traffic incidents, or equipment degradation, delivering resilient performance without a monolithic central controller.

7. Challenges and Future Work

7.1 Scalability and Communication Efficiency

At Internet scale, the bottleneck shifts from client compute to cross-tier bandwidth and aggregator throughput. Even with sparsification and quantization, thousands of heterogeneous clients sending frequent updates can overwhelm message buses and inflate aggregation latency. Hierarchical federation (edge → regional → global), adaptive client selection that prioritizes diversity and novelty, and event-driven rounds that trigger on drift rather than fixed cadences can curb traffic without starving learning. Another open issue is update redundancy: many clients produce near-collinear deltas on stable regimes. Deduplicating or caching “prototype” updates, plus delta-accumulation with error feedback, can slash payloads while preserving signal. Finally, scalable aggregation must be fault-tolerant sharded, stateless workers with idempotent merge semantics to survive partitions and rolling upgrades.

7.2. Privacy Preservation and Security

Gradients and policy deltas can leak sensitive attributes, and open federations invite poisoning or backdoor attacks. Practical privacy requires layered controls: secure aggregation to hide individual updates, per-tenant differential privacy budgets to bound inference risk, and policy distillation or low-rank adapters to minimize exposed parameters. Robustness demands adversarial screening (norm clipping, similarity tests, canary validation) and Byzantine-resilient aggregation when anomalies persist. Keys must rotate automatically; attestations should confirm client integrity (e.g., measured boot) before accepting updates; and audit trails must bind each global model to anonymized participation proofs for compliance. Balancing these safeguards with learning utility especially under strict privacy budgets remains an active trade-off.

7.3. Real-World Deployment Issues

Operationalizing FRL in production surfaces mundane but decisive hurdles: non-IID workload drift breaks lab-tuned hyperparameters; canary environments and progressive rollouts are essential to prevent regressions. Observability must extend to policies versioned artifacts, feature drift monitors, and counterfactual “what-if” replays to build SRE trust. Edge sites face flaky power and intermittent links, so clients need persistent queues, local fallbacks, and safe default policies when global sync stalls. Multi-tenant clusters introduce fairness and resource governance: aggregation should avoid letting high-traffic tenants dominate updates, and scheduling must respect quotas during exploration. Lastly, model lifecycle tasks rollback, hotfixes, and security patches must integrate with existing CI/CD and change-management processes.

7.4. Future Research Directions

Promising avenues include federated curriculum learning that stages tasks from easy to hard across regions to accelerate early convergence, and meta-RL that adapts optimizer and exploration schedules per site automatically. Cross-domain transfer via federated policy distillation could let energy-aware strategies learned in one region bootstrap others with similar carbon intensity profiles. On robustness, combining causal representation learning with robust aggregation may disentangle environment factors from controllable levers, reducing negative transfer. Finally, co-design with systems primitives DPUs/SmartNICs for in-network aggregation, serverless bursts for elastic training, and carbon-aware schedulers can push FRL from promising prototype to default control plane for distributed computing.

8. Conclusion

This work presented a federated reinforcement learning (FRL) framework for decentralized optimization across heterogeneous edge–cloud systems. By training actor–critic agents locally and aggregating only compact model updates with staleness-aware weighting and robustness screens, the approach preserves data locality, reduces control-plane pressure, and avoids single points of failure common to centralized RL. In comprehensive simulations reflecting Kubernetes-style microservice estates with non-IID workloads, churn, and intermittent connectivity, the proposed FRL particularly the asynchronous variant achieved higher SLO compliance and lower tail latency than strong heuristic baselines, while matching the quality of centralized RL without requiring raw trace centralization. Notably, FRL-Async reached ~98.6% SLO compliance with p99 ≈192 ms and lower cost/energy per request, validating its practicality for multi-tenant environments.

Beyond raw performance, the framework demonstrated favorable communication–quality trade-offs through sparsification and quantization, plus operational safeguards such as safety shields, canary checks, and rollback. These results suggest FRL is a credible control plane for real deployments: it adapts quickly to distribution shifts, tolerates partial participation, and scales via hierarchical aggregation. Remaining challenges global scalability under extreme client counts, rigorous privacy budgets with strong utility, and seamless SRE-grade observability define a clear roadmap. Advancing federated curricula, meta-RL for per-site adaptation, and systems co-design with in-network aggregation and carbon-aware scheduling can further harden FRL from prototype to default optimization substrate for distributed computing systems.

References

[1] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. *AISTATS*. https://arxiv.org/abs/1602.05629
[2] Kairouz, P., McMahan, H. B., Avent, B., et al. (2021). Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*. https://arxiv.org/abs/1912.04977
[3] Bonawitz, K., Ivanov, V., Kreuter, B., et al. (2017). Practical Secure Aggregation for Privacy-Preserving Machine Learning. *ACM CCS*. https://arxiv.org/abs/1701.08277
[4] Bonawitz, K., Eichner, H., Grieskamp, W., et al. (2019). Towards Federated Learning at Scale: System Design. *SysML*. https://arxiv.org/abs/1902.01046
[5] Abadi, M., Chu, A., Goodfellow, I., et al. (2016). Deep Learning with Differential Privacy. *ACM CCS*. https://arxiv.org/abs/1607.00133
[6] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*. https://arxiv.org/abs/1908.07873
[7] Li, T., Sahu, A. K., Zaheer, M., et al. (2020). Federated Optimization in Heterogeneous Networks (FedProx). *MLSys Workshop*. https://arxiv.org/abs/1812.06127
[8] Reddi, S. J., Charles, Z., Zaheer, M., et al. (2021). Adaptive Federated Optimization. *ICLR*. https://arxiv.org/abs/2003.00295
[9] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. *NeurIPS*. https://arxiv.org/abs/1703.02757
[10] Yin, D., Chen, Y., Kannan, R., & Bartlett, P. (2018). Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *ICML*. https://arxiv.org/abs/1803.01498
[11] Alistarh, D., Grubic, D., Li, J., Tomioka, R., & Vojnović, M. (2017). QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. *NeurIPS*. https://arxiv.org/abs/1610.02132
[12] Lin, Y., Han, S., Mao, H., Wang, Y., & Dally, W. J. (2018). Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. *ICLR*. https://arxiv.org/abs/1712.01887
[13] Stich, S. U., Cordonnier, J.-B., & Jaggi, M. (2018). Sparsified SGD with Memory. *NeurIPS*. https://arxiv.org/abs/1809.07599
[14] Xie, C., Koyejo, O., & Gupta, I. (2019). Asynchronous Federated Optimization. *arXiv preprint*. https://arxiv.org/abs/1903.03934
[15] Liu, Y., Ding, J., Yang, Y., et al. (2020). A Communication Efficient Vertical Federated Learning Framework. *(See hierarchical/vertical FL concepts)* https://arxiv.org/abs/1912.11187
[16] Nishio, T., & Yonetani, R. (2019). Client Selection for Federated Learning with Heterogeneous Resources. *ICC*. https://arxiv.org/abs/1804.08333
[17] Zhao, Y., Li, M., Lai, L., et al. (2018). Federated Learning with Non-IID Data. *arXiv preprint*. https://arxiv.org/abs/1806.00582
[18] Geyer, R. C., Klein, T., & Nabi, M. (2017). Differentially Private Federated Learning: A Client Level Perspective. *NIPS Workshop*. https://arxiv.org/abs/1712.07557
[19] Fang, M., Cao, X., Jia, J., & Gong, N. Z. (2020). Local Model Poisoning Attacks to Byzantine-Robust FL. *USENIX Security*. https://arxiv.org/abs/1911.11815

[20] Mnih, V., Badia, A. P., Mirza, M., et al. (2016). Asynchronous Methods for Deep Reinforcement Learning. *ICML*. https://arxiv.org/abs/1602.01783
[21] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint*. https://arxiv.org/abs/1707.06347
[22] Konečný, J., McMahan, H. B., Yu, F. X., et al. (2016). Federated Learning: Strategies for Improving Communication Efficiency. *NIPS Workshop*. https://arxiv.org/abs/1610.05492
[23] Satyanarayanan, M. (2017). The Emergence of Edge Computing. *Computer*. https://www.cs.cmu.edu/~satya/docdir/edgecom.pdf
[24] Li, Q., Wen, Z., He, B., et al. (2021). A Survey on Federated Learning Systems: Vision, Hype and Reality. *IEEE TPDS*. https://arxiv.org/abs/1902.04885
[25] Enabling Mission-Critical Communication via VoLTE for Public Safety Networks - Varinder Kumar Sharma - IJAIDR Volume 10, Issue 1, January-June 2019. DOI 10.71097/IJAIDR.v10.i1.1539
[26] Optimizing LTE RAN for High-Density Event Environments: A Case Study from Super Bowl Deployments - Varinder Kumar Sharma - IJAIDR Volume 11, Issue 1, January-June 2020. DOI 10.71097/IJAIDR.v11.i1.1542
[27] Security and Threat Mitigation in 5G Core and RAN Networks - Varinder Kumar Sharma - IJFMR Volume 3, Issue 5, September-October 2021. DOI: https://doi.org/10.36948/ijfmr.2021.v03i05.54992
[28] Kulasekhara Reddy Kotte. 2022. ACCOUNTS PAYABLE AND SUPPLIER RELATIONSHIPS: OPTIMIZING PAYMENT CYCLES TO ENHANCE VENDOR PARTNERSHIPS. International Journal of Advances in Engineering Research , 24(6), PP – 14-24, https://www.ijaer.com/admin/upload/02%20Kulasekhara%20Reddy%20Kotte%2001468.pdf
[29] Naga Surya Teja Thallam. (2022). Enhancing Security in Distributed Systems Using Bastion Hosts, NAT Gateways, and Network ACLs. International Scientific Journal of Engineering and Management, 1(1).
[30] Arpit Garg. (2022). Behavioral biometrics for IoT security: A machine learning framework for smart homes. Journal of Recent Trends in Computer Science and Engineering, 10(2), 71–92. https://doi.org/10.70589/JRTCSE.2022.2.7
[31] Varinder Kumar Sharma - AI-Based Anomaly Detection for 5G Core and RAN Components - International Journal of Scientific Research in Engineering and Management (IJSREM) Volume: 06 Issue: 01 | Jan-2022 .DOI: 10.55041/IJSREM11453
[32] Krishna Chaitanaya Chittoor, "ANOMALY DETECTION IN MEDICAL BILLING USING MACHINE LEARNING ON BIG DATA PIPELINES", INTERNATIONAL JOURNAL OF CURRENT SCIENCE, 12(3), PP-788-796,2022, https://rjpn.org/ijcspub/papers/IJCSP22C1314.pdf
[33] Gopi Chand Vegineni. 2022. Intelligent UI Designs for State Government Applications: Fostering Inclusion without AI and ML, Journal of Advances in Developmental Research, 13(1), PP – 1-13, https://www.ijaidr.com/research-paper.php?id=1454