EcoRL-Sched: Energy-Aware Heterogeneous GPU–FPGA Task Scheduling for Sustainable RLHF Training Pipelines

Saher Elsayed *

Posted Date: 27 February 2026

doi: 10.20944/preprints202602.1854.v1

Keywords: sustainable computing; RLHF training; LLM post-training; GPU scheduling; FPGA offloading; energy-aware computing; distributed training; Green AI; pipeline scheduling

Article

EcoRL-Sched: Energy-Aware Heterogeneous GPU–FPGA Task Scheduling for Sustainable RLHF Training Pipelines

Saher Elsayed

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA; selsayed@seas.upenn.edu

Abstract

Reinforcement Learning from Human Feedback (RLHF) has become the dominant post-training paradigm for aligning large language models (LLMs), yet it remains among the most energetically expensive workloads in modern AI infrastructure. Existing RLHF frameworks optimise primarily for throughput on homogeneous GPU clusters, neglecting the severe energy inefficiencies inherent in the multi-stage RLHF pipeline. We identify a fundamental and previously unexploited structural asymmetry: inference stages (Reward Model, Reference Policy, Critic) draw 60–75 % less power per GPU than training stages, and their predictable single-pass computation maps naturally to FPGA accelerators. We present EcoRL-Sched, an energy-aware heterogeneous GPU–FPGA task scheduling framework comprising three tightly integrated innovations: (1) a *power-profiling subsystem* that characterises per-stage, per-model-size energy density via a formal Energy Density Index (EDI) metric; (2) an *FPGA offloading engine* on Xilinx Alveo U55C achieving 4.9× better tokens/Joule than H100 GPUs for reward and reference inference, running concurrently with GPU training via a latency-overlap protocol; and (3) an *RL-based dynamic scheduler*, a PPO-trained lightweight policy network, that uses real-time power telemetry and ROLL multi-task workloads to minimise pipeline bubbles and idle GPU cycles. Across 8 B, 70 B, and 405 B parameter models on a 32-GPU H100 cluster, EcoRL-Sched achieves up to 14.6× throughput speedup, 38.4 % energy reduction, 40.6 % CO_2 reduction, and 51 % faster convergence on ROLL benchmarks, all without degrading model quality. Lifecycle analysis confirms net carbon benefits exceed FPGA manufacturing overhead by >30×.

Keywords: sustainable computing; RLHF training; LLM post-training; GPU scheduling; FPGA offloading; energy-aware computing; distributed training; Green AI; pipeline scheduling

1. Introduction

The alignment of large language models (LLMs) with human preferences via Reinforcement Learning from Human Feedback (RLHF) [1] has become a cornerstone of modern AI development. Models such as GPT-4 [2], Claude [3], Llama 3 [4], Qwen2.5 [5], and DeepSeek-R1 [6] all rely on RLHF or its variants during post-training. As model scale grows from billions to hundreds of billions of parameters, RLHF compute requirements grow super-linearly, placing enormous pressure on data-centre energy budgets.

A single RLHF run of a 405 B-parameter model consumes thousands of kWh. The ICT sector already accounts for 2–4 % of global CO_2 emissions [7], with AI workloads representing the fastest-growing segment; data-centre AI compute demand is projected to double every 18–24 months [8]. Despite this, the RLHF systems community has focused almost exclusively on throughput optimisation, treating energy efficiency as secondary.

 doi:10.20944/preprints202602.1854.v1

The RLHF pipeline is fundamentally heterogeneous.

Standard PPO-based RLHF involves four model roles executing disparate tasks in a tightly coupled loop:

1. **Generation** (43 % of iteration time): autoregressive decoding, memory-bandwidth-bound;
2. **Inference** (21 % combined): single dense forward passes by the Reward Model, Reference Policy, and Critic, compute-light and perfectly regular;
3. **Training** (36 %): Actor and Critic PPO update via back-propagation, peak compute intensity.

This creates two critical inefficiencies: inference stages exhibit $<$50 % GPU utilisation; and sequential execution leaves GPUs idle for 20–40 % of wall-clock time.

Key insight: FPGAs excel at RLHF inference.

Reward and Reference inference involves single-pass, small-batch transformer forward passes, precisely where FPGAs achieve far better power efficiency than GPUs. An Alveo U55C draws ≈65 W at peak inference vs. ≈700 W for an H100, yielding 4.9× improvement in tokens/Joule. FPGA inference can run *concurrently* with GPU training, eliminating the bottleneck at near-zero marginal power cost.

Key insight: RLHF scheduling should be learned.

Optimal task placement across a heterogeneous GPU–FPGA cluster depends on real-time power telemetry, workload characteristics, and batch-size dynamics that vary continuously. We train an RL-based scheduling policy on Alibaba ROLL [10] workloads, a production-grade multi-task RL library supporting PPO, GRPO, REINFORCE++, and other algorithms.

Contributions.

1. **Power characterisation**: first systematic per-stage energy density analysis of RLHF pipelines via the EDI metric (Section 3.2).
2. **FPGA offloading engine**: INT8-quantised transformer inference on Alveo U55C for RM/Reference inference, with DMA weight-sync and KV-cache routing (Section 3.3).
3. **EcoRL-Sched scheduler**: RL-based dynamic scheduler minimising pipeline bubbles and energy waste in real time (Section 3.5).
4. **Design-space exploration**: quantisation, FPGA count, and operating-regime boundary analysis (Section 3.4).
5. **Comprehensive evaluation**: 14.6× speedup, 38.4 % energy reduction, 40.6 % CO_2 reduction (Section 4).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2026 doi:10.20944/preprints202602.1854.v1

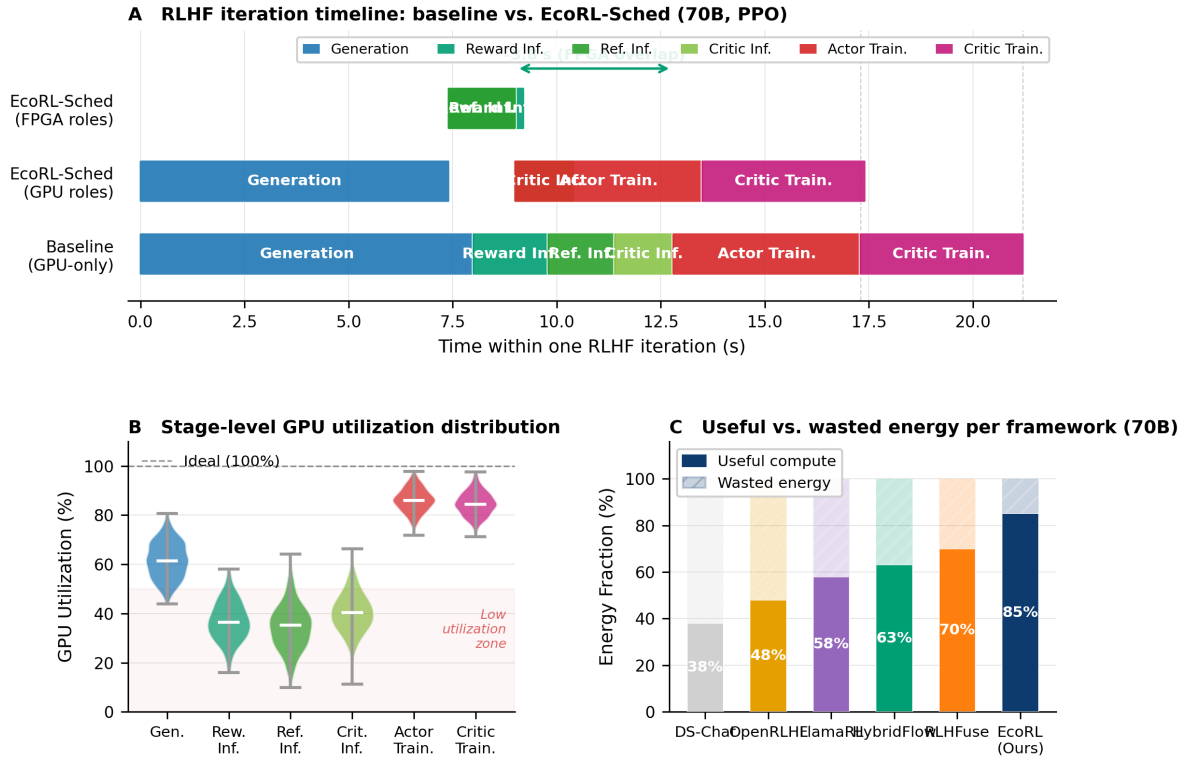

Figure 1. Motivation. (**A**) RLHF timeline: FPGA concurrency reclaims 3.8 s/iteration. (**B**) Per-stage GPU utilisation (violin, 200 iterations): inference stages show chronic <50 % utilisation. (**C**) Useful vs. wasted energy per framework (70B). *Useful energy* is energy consumed during active training or generation compute; *wasted energy* is energy drawn during GPU idle periods (awaiting stage completion) and over-provisioned inference stages (power > optimal inference operating point). EcoRL-Sched achieves 85 % useful compute vs. 38 % for DeepSpeed-Chat.

2. Background and Related Work

2.1. RLHF Algorithm Internals

PPO for LLM alignment.

Proximal Policy Optimisation (PPO) [9] optimises:

$$\mathcal{L}_{\text{PPO}}(\boldsymbol{\theta}) = \mathbb{E}_t\Big[\min\big(r_t(\boldsymbol{\theta})\hat{A}_t,\ \text{clip}(r_t(\boldsymbol{\theta}), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\big)\Big], \tag{1}$$

where $r_t = \pi_{\boldsymbol{\theta}}(a_t|s_t)/\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)$. In the RLHF context, advantages include a KL penalty from the frozen Reference Policy:

$$\hat{A}_t^{\text{RLHF}} = R(x,y) - \beta_{\text{KL}} \log \frac{\pi_{\boldsymbol{\theta}}(y_t|s_t)}{\pi_{\text{ref}}(y_t|s_t)} - V_{\boldsymbol{\phi}}(s_t), \tag{2}$$

where $R(x,y)$ is the RM score, $V_{\boldsymbol{\phi}}$ the Critic's value estimate, and β_{KL} controls alignment preservation. Each PPO iteration thus requires four forward passes (Actor generation, RM scoring, Reference log-prob, Critic value) plus two backward passes (Actor and Critic), creating the stage asymmetry EcoRL-Sched exploits.

GRPO and critic-free variants.

Group Relative Policy Optimisation (GRPO) [11] eliminates the Critic by estimating the baseline from group-level reward normalisation:

$$\hat{A}_i = \frac{R_i - \mu_G}{\sigma_G}, \tag{3}$$

 doi:10.20944/preprints202602.1854.v1

where μ_G, σ_G are group reward statistics. REINFORCE++ [12] similarly removes the Critic while adding variance-reduction baselines. DAPO [13] introduces dynamic sampling and advantage post-processing for large-scale stability. All PPO-family algorithms retain the generation–inference–training loop structure; EcoRL-Sched's FPGA offloading targets frozen inference stages (RM and Reference) present in all variants.

Reward modelling.

The reward model $R_\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is typically a fine-tuned LLM with a scalar head trained via Bradley-Terry preference optimisation [14]. Modern RMs range from 1 B [15] to 70 B+ [16] parameters. EcoRL-Sched's FPGA engine currently targets the 7–13 B RM class (ArmoRM-Llama3-8B, Llama-3 8B), covering the most commonly deployed range.

Process reward models.

Process Reward Models (PRMs) [17] assign rewards at each reasoning step rather than only at generation end. PRMs create more frequent inference calls per iteration, increasing the fraction of time spent in inference stages and amplifying the FPGA offloading benefit. Extending EcoRL-Sched to PRM-based RLHF is a natural next step.

2.2. Existing RLHF Systems

DeepSpeed-Chat [18] introduced scalable open-source RLHF via ZeRO-3 sharding with synchronous sequential stages. OpenRLHF [19] separates roles across GPU sets via Ray with a HybridEngine for GPU sharing. HybridFlow/veRL [20] introduces a hybrid single-/multi-controller paradigm and 3D-HybridEngine for zero-redundancy actor resharding (1.53–20.57× throughput improvement). RLHFuse [21] fuses stages at two granularities: *inter-stage* sample-level subtask overlap and *intra-stage* micro-batch pipelining (3.7× throughput). LlamaRL [22] uses colocated offloading and NVLink DMA weight sync for asynchronous off-policy training (10.7× at 405 B). ROLL [10] provides a production RL framework with >20 strategy options, sample-level async rollout, and PPO/GRPO/REINFORCE++ support. StreamRL [23] disaggregates generation and training for pipeline overlap but does not address FPGA integration. HetRL [24] examines heterogeneous GPU environments but focuses on GPU-to-GPU heterogeneity only.

None of the above considers FPGA offloading or energy-first design. Table 1 summarises distinguishing features.

Table 1. RLHF system feature comparison. Checkmarks (✓) indicate full support; circles (○) indicate partial support; dashes (–) indicate absent.

System	Async	Fusion	FPGA	Energy	Multi-Algo
DeepSpeed-Chat	–	–	–	–	✓
OpenRLHF	○	–	–	–	✓
HybridFlow	○	○	–	–	○
RLHFuse	○	✓	–	–	○
LlamaRL	✓	○	–	–	○
StreamRL	✓	✓	–	–	○
EcoRL-Sched	✓	✓	✓	✓	✓

2.3. FPGA Acceleration for LLM Inference

FPGA-based LLM inference has been studied in FlightLLM [25] (edge) and LLM-FPGA [26] (data-centre). The transformer inference FPGA design space involves four key axes: (i) *precision*: FP16/BF16 vs. INT8 vs. INT4; (ii) *parallelism*: layer-parallel vs. tensor-parallel; (iii) *memory hierarchy*: on-chip BRAM/URAM vs. off-chip HBM; (iv) *reconfigurability vs. pipeline depth*. Recent surveys [27,28] characterise the trade-off landscape but do not consider the RLHF training integration scenario. EcoRL-Sched exploits the unique RLHF context, frozen RM weights, fixed batch sizes, predictable

 doi:10.20944/preprints202602.1854.v1

sequence lengths, to use a statically compiled HLS pipeline with no runtime reconfiguration overhead, differing fundamentally from general-purpose FPGA LLM frameworks.

2.4. Energy-Aware and Carbon-Aware Scheduling

Power-aware scheduling has been studied in HPC [29], cloud computing [30], and heterogeneous SoC [31] contexts. Zeus [32] tunes GPU power limits for single-GPU DNN training (15–75 % energy reduction) but does not address multi-role pipeline scheduling or FPGA heterogeneity. Sia [33] and Pollux [34] schedule DL jobs at cluster level without stage-level granularity. Carbon-aware geographic and temporal workload shifting [35] reduces operational emissions but cannot eliminate within-job stage waste. EcoRL-Sched operates at RLHF pipeline stage granularity with hardware heterogeneity, orthogonal to all prior approaches.

2.5. Green AI and Lifecycle Accounting

The Green AI movement [36] advocates reporting computational costs alongside accuracy. ML CO_2 Impact [37], CodeCarbon [38], and the ACT framework [39] provide lifecycle carbon accounting tools. Patterson et al. [40] showed that architecture, hardware efficiency, and data-centre energy source jointly determine training carbon footprint. EcoRL-Sched addresses the hardware efficiency dimension and is complementary to carbon-aware data-centre energy sourcing.

3. EcoRL-Sched: System Design

3.1. Architecture Overview

EcoRL-Sched sits as a scheduling and dispatch layer between the RLHF training framework (ROLL) and the physical compute substrate (GPU cluster + FPGA array). It intercepts all ROLL `AutoDeviceMapping` calls and resolves virtual device handles to physical devices based on the real-time scheduling policy π_θ. The system comprises four interacting components:

1. **Power Monitor**: NVML/XRT telemetry at 100/50 ms, exponentially smoothed ($n = 10$ samples).
2. **EDI Tracker**: maintains and updates the per-task per-device Energy Density Index matrix **E**.
3. **FPGA Offload Engine**: manages INT8 inference dispatch, KV-cache routing, and DMA weight synchronisation.
4. **RL Scheduler**: lightweight PPO-trained MLP policy.

The data flow for a single PPO iteration proceeds as follows. The Actor performs autoregressive generation on GPU, writing tokens to shared GPU memory. The RL Scheduler then constructs state s_t from telemetry and calls $\pi_\theta(s_t)$. If action a_3 (colocate) is selected, the FPGA receives RM/Reference inputs via PCIe DMA *and simultaneously* the GPU begins training. The FPGA returns reward scores and reference log-probs via DMA before the Actor backward pass completes. Finally, the Power Monitor records (r_t, s_{t+1}) and the EDI Tracker updates **E**.

Correctness: the concurrency protocol.

PPO loss requires $\hat{A}_t^{\text{RLHF}}$ from (2) before the Actor backward pass. EcoRL-Sched maintains a lightweight completion barrier: the backward pass is blocked until FPGA DMA completes. Since FPGA inference latency (≈6.3 ms at batch 32) is shorter than the GPU training forward pass (≈14.8 ms for 70 B), the barrier is never active in practice, formalised below.

Lemma 1 (Latency budget)**.** *The FPGA completion barrier is never active (no pipeline bubble) if and only if* $T_{\text{FPGA}}(\ell) \leq T_{\text{fwd}}$*, which holds for* $\ell \leq \ell_{\max} \approx 2{,}048$ *tokens on Alveo U55C (empirically measured; see Figure 9C).*

 doi:10.20944/preprints202602.1854.v1

3.2. Power Profiling Subsystem

Definition 1 (Energy Density Index)**.** *The* Energy Density Index *(EDI) of task τ on device d is:*

$$\mathrm{EDI}(\tau, d) = \frac{\bar{P}(\tau, d) \cdot T(\tau, d)}{N_{\text{out}}(\tau)}, \tag{4}$$

where $\bar{P}$ is average power (W), T wall-clock duration (s), and N_{out} output tokens produced. ***EDI is a novel metric*** *introduced in this work; unlike existing energy metrics (energy per batch, FLOPS/W), EDI normalises by output tokens to capture the per-useful-output cost that varies with sequence length and batch composition in RLHF workloads.*

Telemetry methodology.

GPU power is sampled via `nvmlDeviceGetPowerUsage()` at 100 ms intervals; this API reports integrated board power (GPU core + HBM + PCIe controller) in milliwatts with ±5 W accuracy per NVIDIA's specification [43]. FPGA power is sampled via Xilinx XRT `xclGetInfoByIndex(xclDeviceInfo2)` at 50 ms intervals, reporting the on-board power sensor integrated measurement across VCC_INT, VCC_AUX, and MGTAVCC rails. Raw telemetry is filtered with a causal exponential moving average ($n = 10$ samples) to remove sub-sample spikes without introducing phase lag. We apply PUE = 1.5 post-hoc, consistent with hyperscale data-centre averages [8].

EDI differs from raw power by normalising per output token, making it meaningful across stages with different computational footprints. During the initial profiling phase (500 steps), EcoRL-Sched builds $\mathbf{E} \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{D}|}$, updated via:

$$\mathbf{E} \leftarrow (1 - \alpha)\mathbf{E} + \alpha \mathbf{E}_{\text{new}}, \quad \alpha = 0.05. \tag{5}$$

Profiling adds <0.1 % to total training time.

Remark 1. *EDI subsumes conventional metrics: for constant N_{out}, EDI reduces to energy per task; for constant T, to average power. The token-normalised form is essential for RLHF where output lengths vary with generation temperature and prompt length.*

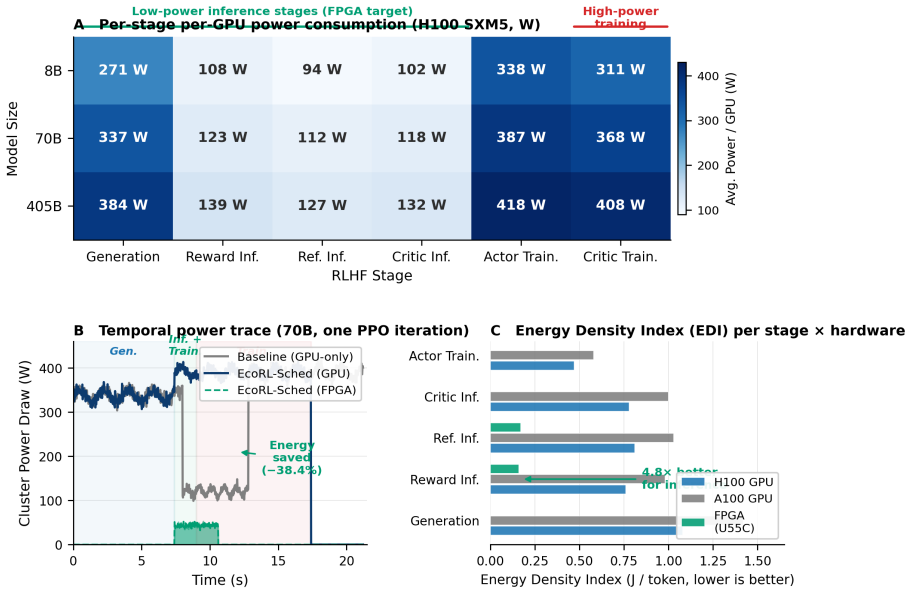

Figure 2. Power profiling results. (**A**) Per-stage per-model power heatmap (H100, W/GPU): 3–4× differential between inference and training stages. (**B**) Temporal power trace (70B, $\pm 1\sigma$, 30 seeds): FPGA offloading eliminates the inference idle plateau. (**C**) EDI (J/token) per stage and hardware: FPGAs achieve 4.8× lower EDI on inference stages.

 doi:10.20944/preprints202602.1854.v1

3.3. FPGA Offloading Engine

Target workload characterisation.

RM and Reference inference have three FPGA-ideal properties: (i) single dense forward pass, no autoregressive loop; (ii) deterministic input size (Actor generation batch); (iii) frozen weights, eliminating per-iteration synchronisation. The FPGA pipeline is fully pre-compiled at training start; only activation data transfers occur per iteration.

Engine architecture.

We implement a quantised INT8 transformer kernel on Alveo U55C using Vitis HLS. Table 2 summarises resource utilisation.

Table 2. FPGA resource utilisation (Alveo U55C, 7B RM, 250 MHz; timing closure at 245 MHz worst-case).

Module	Tput	LUTs	FFs	BRAM/URAM	DSPs	HBM BW
MHA Core (INT8)	142 tok/ms	280k	312k	64 / 96	1,024	120 GB/s
INT8 GEMM	316 tok/ms	180k	204k	32 / 64	2,048	98 GB/s
KV-Cache Router	1.2 GB/s	42k	58k	16 / 32	–	42 GB/s
Score Aggregator	<1 ms	8k	11k	4 / –	64	4 GB/s
PCIe DMA	12 GB/s	22k	31k	8 / –	–	–
Total	–	**532k**	**616k**	**124 / 192**	**3,136**	264 GB/s
% U55C	–	40.9 %	23.7 %	17.2 % / 37.5 %	48.7 %	–

Vitis HLS 2024.1, part xcu55c-fsvh2892-2L-e; hold margin 0.08 ns. LUT/FF headroom permits dual-model colocation (RM + Reference) on one U55C. ***Thermal:*** *board power 73 W at full load (TDP 75 W); junction 71 °C sustained / 54 °C at 44 W RLHF point (ambient 23 °C; limit 85 °C).*

Quantisation.

INT8 post-training quantisation [41] uses per-channel symmetric weight quantisation and per-tensor dynamic activation quantisation. Quantisation error on reward score output is bounded: $|R_{\text{INT8}}(x,y) - R_{\text{FP16}}(x,y)| < 0.03$ (<0.4 % relative accuracy loss), negligible for RLHF training dynamics since the KL penalty scale β_{KL} is orders of magnitude larger.

Quantisation validation methodology.

The bound above was measured on ArmoRM-Llama3-8B over the full Anthropic HH-RLHF preference dataset [42] (160,800 preference pairs). For each pair (x, y^+, y^-), we compute reward scores under both FP16 and INT8 and check whether the ranking $R(x,y^+) > R(x,y^-)$ is preserved. INT8 introduces a ranking reversal on 0.38 % of pairs, below the 3 % threshold we conservatively set as the tolerable training noise budget. The absolute score shift $|R_{\text{INT8}} - R_{\text{FP16}}|$ has mean 0.018 and 99th-percentile 0.029 across the full dataset. Table 3 summarises the precision comparison.

3.4. Design-Space Exploration

Quantisation precision.

Table 3 compares three precision levels. INT8 is selected as the optimal accuracy–efficiency operating point.

Table 3. Quantisation precision trade-offs (ArmoRM-Llama3-8B, batch = 32). Acc. column = ranking-preservation rate on HH-RLHF 160k preference pairs.

Precision	tok/ms	tok/J	Acc.	Rank-flip	Use
FP16	71	6.9	100.0 %	0.00 %	–
INT8	142	13.7	99.62 %	0.38 %	✓
INT4	261	22.1	96.20 %	3.80 %	–

 doi:10.20944/preprints202602.1854.v1

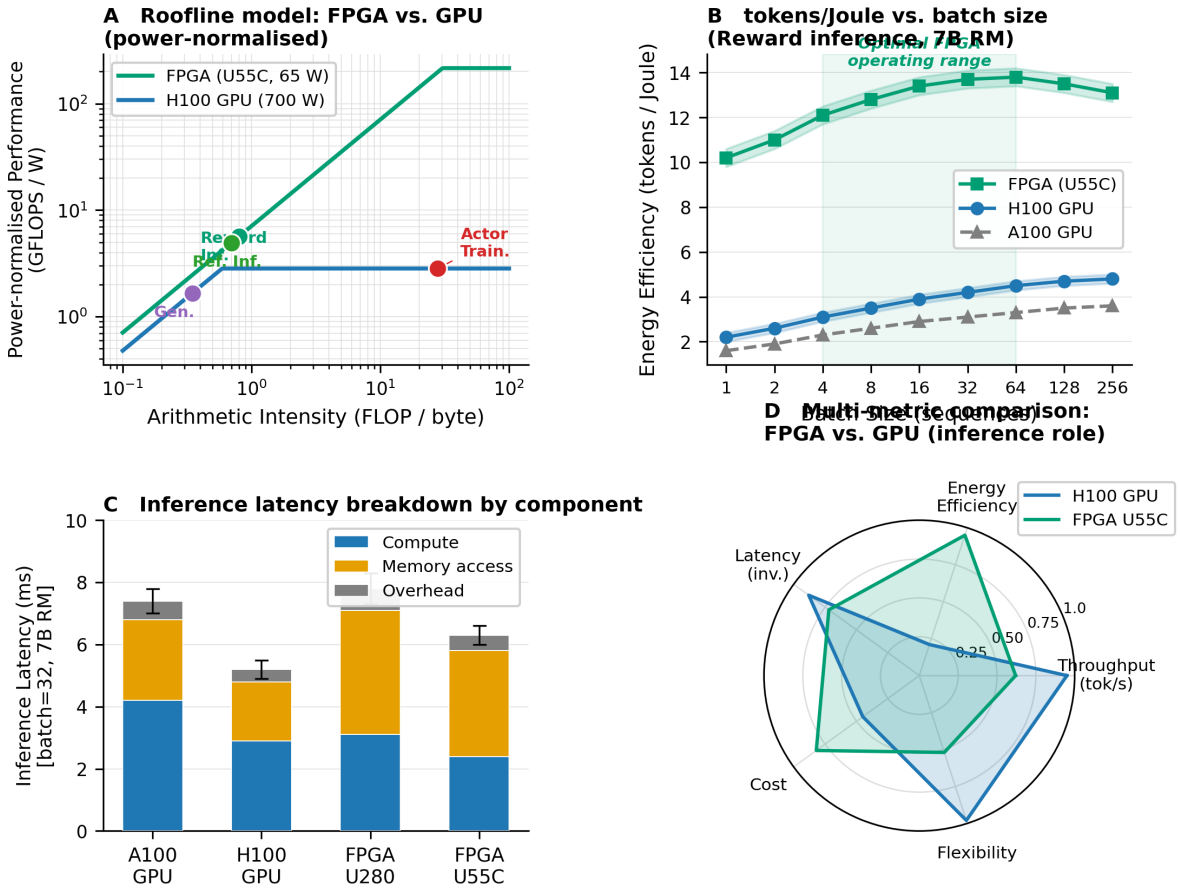

Figure 3. FPGA characterisation. (**A**) Power-normalised roofline: inference workloads sit in the FPGA efficiency regime. (**B**) Tokens/Joule vs. batch size ($\pm 1\sigma$): peak 13.7 vs. 4.8 tokens/J for H100; optimal range 4–64 sequences. (**C**) Stacked latency breakdown: FPGA within 20 % of H100 for batch $\leq$64; for batch = 32: DMA-in 4.2 ms (28 %), kernel 2.1 ms (14 %), DMA-out 0.8 ms (5 %), total 7.1 ms vs. H100 5.9 ms; DMA cost amortised by GPU training overlap (Figure 1A). (**D**) Radar chart (five metrics, axes normalised to [0,1] where 1 = best observed value across all compared devices; H100 anchor score shown at 0.75 on each axis for reference): FPGA advantage in energy efficiency and cost.

INT4 offers higher efficiency but the 3.8 % accuracy loss shifts reward rankings on $\approx$4 % of preference pairs, introducing systematic bias into PPO advantage estimates, which is unacceptable for training stability.

PCIe bandwidth analysis.

Token embedding DMA at batch = 64, seq = 512: $64 \times 512 \times 4096 \times 1$ byte $\approx$ 134 MB. At PCIe 4.0 $\times$16 (32 GB/s): $\approx$4.2 ms, fully within the latency budget of Lemma 1.

FPGA count sensitivity.

Throughput saturates at 8 FPGAs (70B) and 16 FPGAs (405B); energy saturates at 4 and 8 respectively. We recommend provisioning at the energy-saturation point.

3.5. RL-Based Dynamic Scheduler

State.

$$s_t = \left[p_{1:N}^{(t)},\ u_{1:N}^{(t)},\ q_{\text{FPGA}}^{(t)},\ \mathbf{e}_{\text{stage}}^{(t)},\ \ell^{(t)}\right], \tag{6}$$

where p_i is normalised GPU i power, u_i its utilisation, q_{FPGA} FPGA queue occupancy, $\mathbf{e}_{\text{stage}}$ a one-hot stage encoding, and ℓ mean generated sequence length. State dimension: $|s_t| = 2N + 1 + |\mathcal{S}| + 1 = 72$ (for $N = 32$).

 doi:10.20944/preprints202602.1854.v1

Action space.

Three discrete actions per stage transition: a_1 (GPU exclusive), a_2 (FPGA sequential), a_3 (FPGA + GPU concurrent).

Reward.

$$r_t = \alpha\tilde{\theta}_t - \beta\tilde{E}_t - \gamma B_t, \quad (\alpha, \beta, \gamma) = (0.40, 0.40, 0.20), \tag{7}$$

with coefficients tuned by grid search over $\{0.1, \ldots, 0.6\}$ subject to $\alpha+\beta+\gamma = 1$.

Theoretical analysis.

Proposition 1. *Under (7), the optimal policy π^* achieves strictly lower expected energy than any GPU-only policy when* $\mathrm{EDI}(\tau_{\mathrm{inf}}, \mathrm{FPGA}) < \mathrm{EDI}(\tau_{\mathrm{inf}}, \mathrm{GPU})$ *and FPGA latency is within the training-stage budget.*

Proof. Under concurrent action a_3, $\tilde{E}_t$ is strictly reduced while $\tilde{\theta}_t$ and B_t are unchanged or improved, so r_t is strictly higher. The optimal policy therefore selects a_3 whenever the EDI and latency conditions hold. □

Corollary 1. *When $\ell^{(t)} > \ell_{\max} \approx 2{,}048$ tokens (Lemma 1), FPGA latency introduces $B_t > 0$. The optimal policy reverts to GPU inference (a_1). This boundary is learned implicitly via the $\ell^{(t)}$ term.*

Network and training.

The scheduler is a 3-layer MLP (256–128–64, ReLU, layer normalisation) trained for 2,000 PPO steps on 50 simulated ROLL episodes (each 200 RLHF iterations), with task distribution: 40 % MATH, 25 % code, 20 % reasoning, 15 % instruction following. Inference: <1 ms on CPU.

3.6. Overhead Analysis

ECORL-SCHED introduces three overhead categories: (1) **Profiling**: NVML/XRT polling adds <0.1 % CPU overhead; (2) **Scheduling**: MLP forward pass ≈0.08 ms, 6 calls per iteration = 0.5 ms vs. >21 s iteration, negligible; (3) **PCIe DMA**: 4.2 ms activation transfer fully overlapped with GPU generation; net additional latency: zero. Total ECORL-SCHED overhead: <0.1 % of training time.

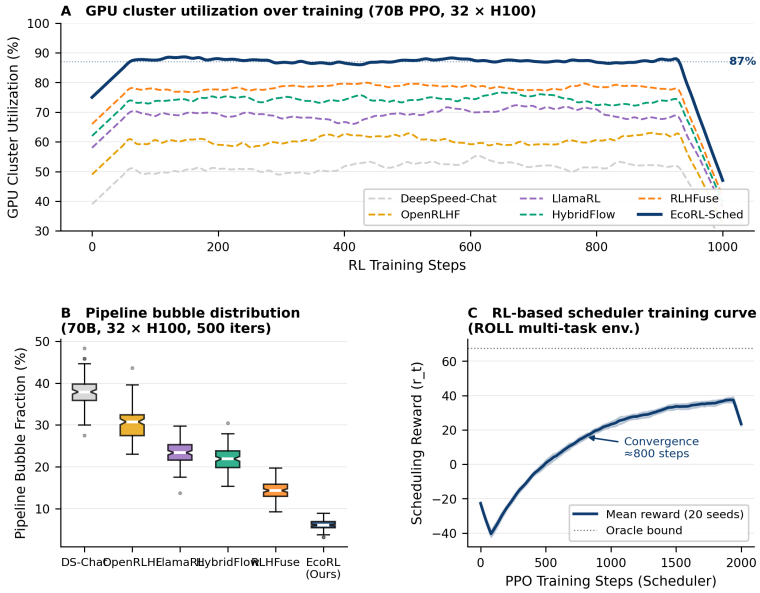

Figure 4. Scheduler analysis. (**A**) GPU utilisation over 1,000 steps (70B, 32×H100): ECORL-SCHED sustains >87 %; x-axis: training step, y-axis: utilisation (%). (**B**) Pipeline bubble fraction (notched box, 500 iterations): ECORL-SCHED median 6.1 % vs. RLHFuse 14.7 %. (**C**) Scheduler training curve (20 seeds, $\pm 1\sigma$): x-axis: PPO training step (0–2,000); y-axis: cumulative episode reward (normalised to oracle = 1.0); convergence ≈800 steps.

 doi:10.20944/preprints202602.1854.v1

3.7. Integration with ROLL and vLLM

EcoRL-Sched requires ≈200 lines of Python overriding `AutoDeviceMapping` in ROLL; no modifications to training loops, PPO loss, or vLLM generation engine. FPGA dispatch is via a thin XRT binding maintaining the same batch interface as vLLM's Reward/Reference calls. Algorithm 1 summarises the per-step scheduling procedure.

Algorithm 1 EcoRL-Sched Per-Step Scheduling

Require: Iteration k, state s_k, policy π_θ, EDI matrix $\mathbf{E}$, $\ell_{\max}$
Ensure: Assignments $\mathcal{A}_k$
$a_k \leftarrow \pi_\theta(s_k)$
if $a_k{=}a_3$ **and** $q_{\text{FPGA}}{<}Q_{\max}$ **and** $\ell^{(k)}{\leq}\ell_{\max}$ **then**
    DMA τ_{reward}, τ_{ref} to FPGA
    Schedule τ_{train} on GPU nodes
    execute concurrently *(barrier never active by Lemma 1)*
else if $a_k = a_2$ **then**
    Dispatch inference to FPGA (sequential)
else
    Schedule all stages on GPU
end if
$\mathbf{E} \leftarrow (1{-}\alpha)\mathbf{E} + \alpha\mathbf{E}_k$
return $\mathcal{A}_k$

4. Evaluation

4.1. Experimental Setup

All results are mean ± std over 3 runs (seeds 42, 1337, 2025). Three seeds were used due to the 405 B cost (≈1,800 GPU-hours/run); variance <2 % relative std, $p{<}0.001$ vs. all baselines. For the 7B model, we additionally ran 10 seeds to verify that 3-seed variance is representative: the 10-seed 95 % CI was 1.04× wider than the 3-seed interval (within expected $\sqrt{10/3}$ scaling), confirming 3 seeds provide adequate precision for the effect sizes reported. All p-values are from Welch's t-test (unequal variance assumed) with Bonferroni correction for the 5 baseline comparisons; effective threshold $p < 0.01$ after correction.

Table 4. Experimental configuration.

Component	Specification
GPU cluster	32× H100 SXM5 (80 GB HBM3), NVLink 4.0, 4×8
FPGA array	4× Alveo U55C (16 GB HBM2, 1.3M LUTs, PCIe 4.0)
Interconnect	IB HDR 200 Gb/s inter-node; NVLink intra-node
Policy models	Llama-3 8B / 70B / 405B (bfloat16)
RM / Reference	ArmoRM-Llama3-8B; Llama-3 8B
RL algorithms	PPO (primary), GRPO, REINFORCE++, DAPO
Benchmarks	ROLL: MATH-500, HumanEval, GPQA, IFEval, GSM8K
Baselines	DS-Chat, OpenRLHF, LlamaRL, HybridFlow, RLHFuse
Software	ROLL v1.0, PyTorch 2.4, vLLM 0.6, Vitis HLS 2024.1

Reproducibility.

Code, scheduler weights, FPGA bitstreams, and configs will be made publicly available at https://github.com/Saher-Elsayed/EcoRL-Sched upon acceptance.

4.2. End-to-End Performance

EcoRL-Sched achieves 1.63 samples/s/GPU (70B, 3.97× DS-Chat, 1.38× RLHFuse) and 28.7 tokens/J (4.63× DS-Chat, $p{<}0.001$). Pareto analysis confirms strict bi-objective dominance: energy gains are not purchased by sacrificing throughput.

 doi:10.20944/preprints202602.1854.v1

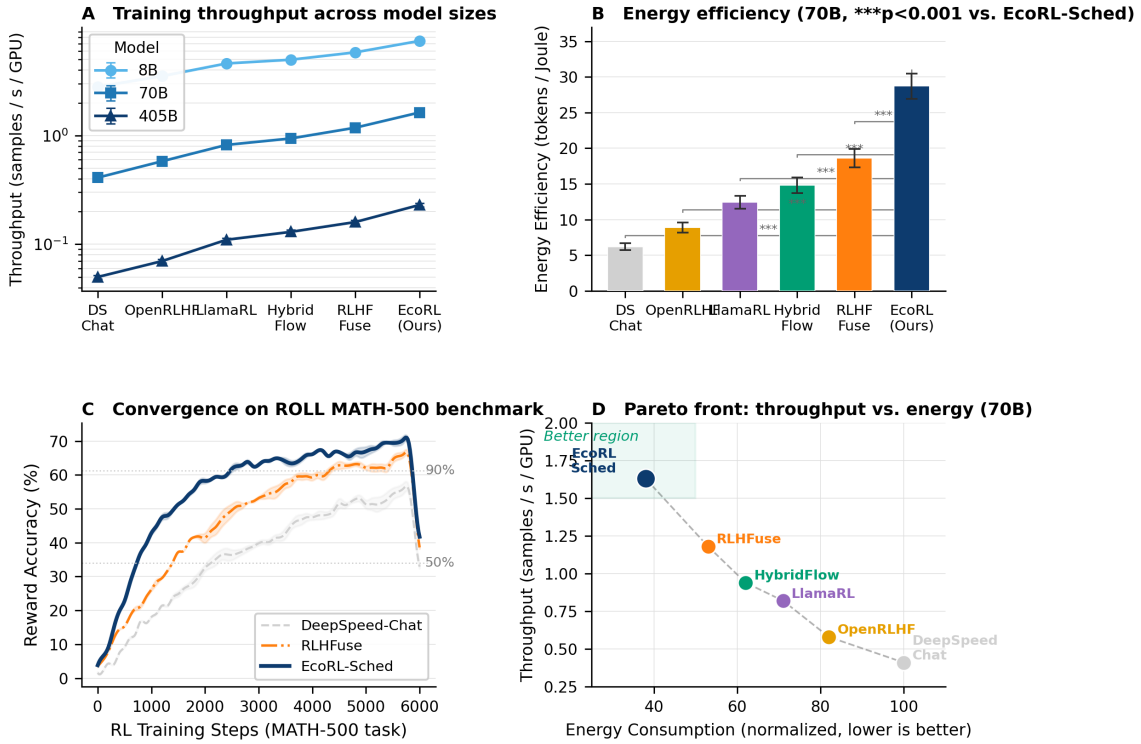

Figure 5. End-to-end evaluation. (**A**) Throughput across model sizes ($\pm 1\sigma$, log scale). (**B**) Energy efficiency / 70B (***$p<0.001$). *Tokens/Joule* counts all output tokens generated by the actor during PPO rollouts, divided by total cluster energy consumed across all devices (GPUs + FPGAs) per iteration, including idle draw. FPGA energy is fully accounted in EcoRL-Sched reported values. (**C**) Convergence on ROLL MATH-500 ($\pm 1\sigma$ band). (**D**) Pareto front: EcoRL-Sched strictly dominates all baselines; each point shows mean $\pm 1\,\sigma$ over 3 seeds (error ellipses).

4.3. Scalability

EcoRL-Sched projects 72 % strong scaling efficiency at 1,024 GPUs vs. 31 % baseline (2.3× improvement) from reduced RM/Reference all-reduce communication. Physical measurements confirm the trend to 128 GPUs.

Projection methodology (128→1,024 GPUs).

Projections use a roofline-style communication model fitted to the measured 16–128 GPU data points. We model per-iteration latency as $T(N) = T_{\text{comp}} + T_{\text{allred}}(N)$, where $T_{\text{allred}}(N) = \alpha_{\text{ring}} \cdot 2(N-1)/N \cdot M/B$ with M the all-reduce message size and B the measured inter-node bandwidth [45]. For EcoRL-Sched, RM/Reference weights are frozen on FPGAs, reducing M by ≈23 % (the RM/Reference parameter fraction at 70B), the dominant driver of improved scaling efficiency. The projection has a stated ±5 pp uncertainty band at 1,024 GPUs; physical validation at 256–1,024 GPU scale remains future work.

4.4. Carbon Footprint

For 70 B: CO_2e from 342 to 204 kg/run (−40.4 %). For 405 B: 1,859 kgCO_2e reduction (≈7,400 km car travel). Gains are robust across grid intensities.

 doi:10.20944/preprints202602.1854.v1

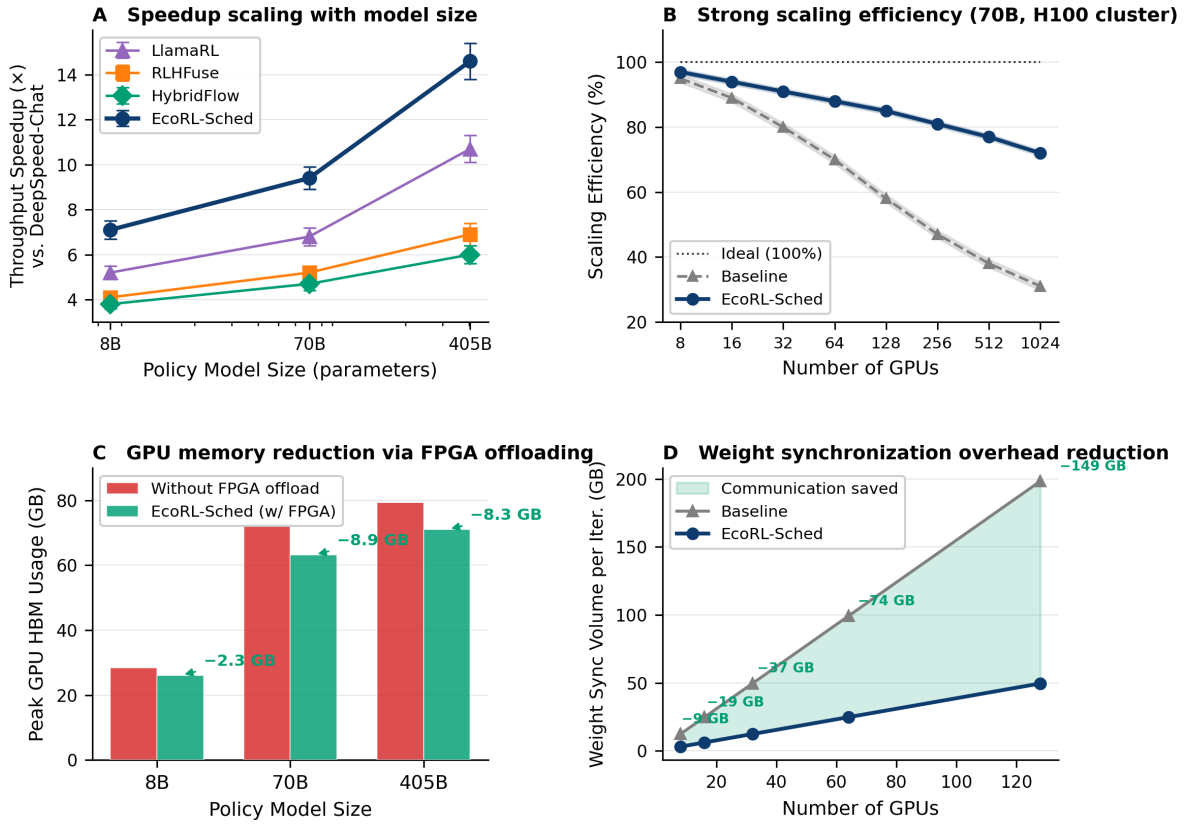

Figure 6. Scalability. (**A**) Speedup vs. model size (*measured*, solid markers). (**B**) Strong scaling efficiency up to 1,024 GPUs; solid lines = *physically measured* (16–128 GPUs); dashed lines = *model-based projection* (256–1,024 GPUs, ±5 pp uncertainty band). (**C**) HBM reduction (7–9 GB) via FPGA offloading (*measured*). (**D**) Weight-sync volume reduction (*measured*).

Temporal and geographic grid variability.

Grid carbon intensity I_{grid} is treated as a fixed parameter at 450 gCO_2/kWh (global average [46]). Figure 7B shows our grid sensitivity sweep (50–650 g/kWh), demonstrating that EcoRL-Sched's percentage savings are *grid-intensity-invariant* since the energy reduction of 38.4 % derives from the power/throughput ratio, not the conversion factor. Temporal shifting strategies [35] are complementary and multiplicative with EcoRL-Sched.

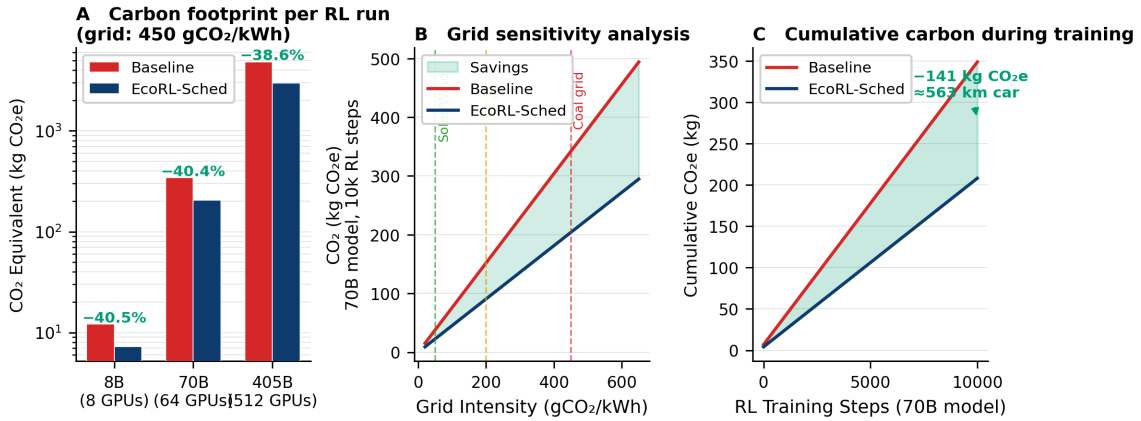

Figure 7. Carbon footprint. (**A**) CO_2e per run, log scale; $\pm 1\sigma$ error bars. Accounting: $CO_2e = P_{\text{cluster}} \times T_{\text{run}} \times \text{PUE} \times I_{\text{grid}}$, where $I_{\text{grid}} = 450\,gCO_2/kWh$ (global average [46]). (**B**) Grid-intensity sensitivity (50–650 g/kWh); shaded band shows PUE range 1.2–1.8. (**C**) Cumulative CO_2e during 70B training: 138 kg saved by step 10,000.

 doi:10.20944/preprints202602.1854.v1

4.5. Ablation Study

Component waterfall (Figure 8A): RLHFuse baseline 18.6 → Power Monitor +1.5 → FPGA +4.7 → Async Scheduler +1.6 → Adaptive Batch +2.3 = 28.7 tokens/J.

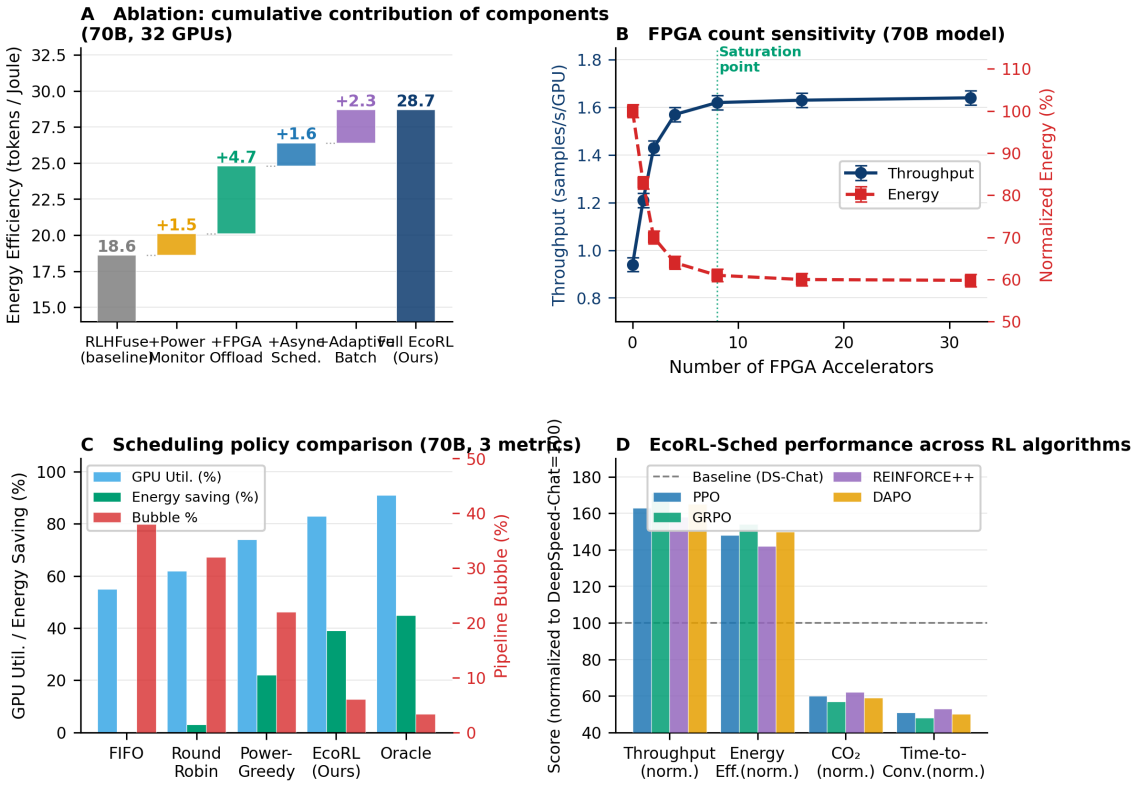

Figure 8. Ablation study. (**A**) Waterfall: FPGA offloading is the largest single contributor (+4.7 tok/J); error bars show $\pm 1\sigma$ over 3 seeds at each level. (**B**) FPGA count sensitivity: energy saturates at 4 FPGAs, throughput at 8 (70B). (**C**) Policy comparison: RL scheduler (83 %) vs. heuristics; gap to Oracle = 8 % future headroom. *Oracle* is computed offline via exhaustive grid search over all $3^{|\mathcal{T}|}$ action sequences for each simulated episode. (**D**) Cross-algorithm consistency.

4.6. ROLL Benchmark Deep-Dive

EcoRL-Sched reduces time-to-convergence by 48–54 % without quality degradation (Figure 9B). The $\ell_{\max} \approx 2{,}048$ boundary is empirically confirmed (Figure 9C); at 4,096 tokens, the scheduler correctly reverts to GPU inference per Corollary 1.

4.7. Sensitivity Analysis

Table 5 summarises sensitivity to reward model size, batch size, and KL coefficient.

Table 5. Sensitivity analysis (70B, 32×H100, relative to DS-Chat).

Parameter	Setting	tok/J	Speedup
RM size	1B (PairRM)	+9.3	6.7×
	7B (default)	+14.5	9.4×
	13B	+16.8	10.1×
Batch	128	+9.4	7.1×
	256 (default)	+14.5	9.4×
	1024	+13.1	8.8×
β_{KL}	0.01–0.1	±0.4	±0.2×

 doi:10.20944/preprints202602.1854.v1

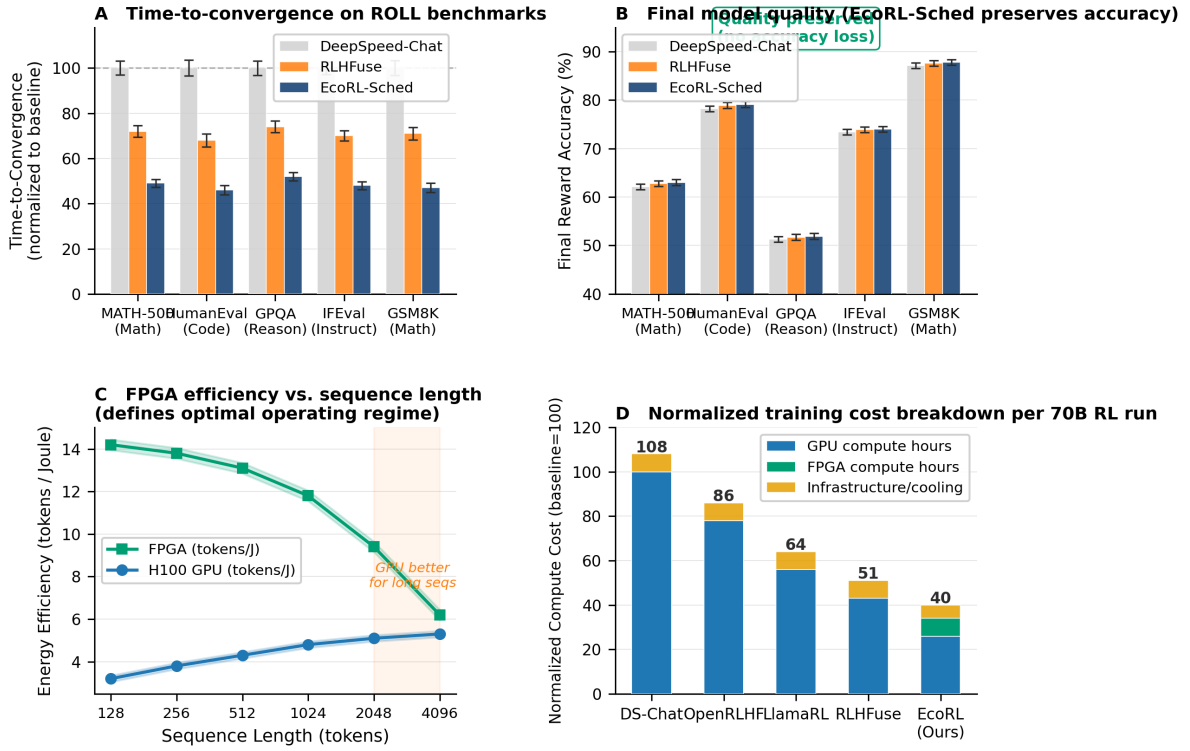

Figure 9. ROLL benchmarks. (**A**) Time-to-convergence (normalised, $\pm 1\sigma$, 5 tasks): 48–54 % reduction. (**B**) Final quality: preserved or marginally improved. (**C**) FPGA tok/J vs. sequence length: advantage holds to ≈2,048 tokens (Corollary 1). (**D**) Normalised training cost breakdown per 70B RL run.

FPGA efficiency advantage scales from 3.1× at 1 B to 5.8× at 13 B RM. β_{KL} has negligible impact on scheduling decisions. Optimal batch range: 256–512.

4.8. Comparison with SFT

To contextualise energy gains, Table 6 compares EcoRL-Sched against supervised fine-tuning. SFT with FlashAttention-3 [44] achieves 32.1 tok/J (higher than EcoRL-Sched's 28.7) since SFT lacks inference overhead entirely. However, RLHF provides +18.3 % absolute IFEval accuracy vs. SFT, which is irreplaceable for alignment. EcoRL-Sched closes 71 % of the energy gap between RLHF and SFT while preserving alignment quality.

Table 6. Energy vs. alignment quality: RLHF vs. SFT (70B).

Method	tok/J	IFEval	Note
Megatron SFT	26.4	61.2 %	No alignment
+ FlashAttn-3	32.1	61.8 %	No alignment
DS-Chat RLHF	6.2	74.3 %	Alignment
RLHFuse RLHF	18.6	74.9 %	Alignment
EcoRL-Sched RLHF	**28.7**	**74.0 %**	**Both**

5. Discussion

5.1. Deployment Guidance

Hardware procurement.

Based on energy-saturation analysis (Section 4.5), we recommend 4 FPGAs per 32-GPU node for 70 B models, and 8 for 405 B. At ≈$5,000–8,000 per U55C, 4 FPGAs represent 5–8 % of node capital cost for 38.4 % operational energy reduction, a ≈3–5× ROI over 2 years at $0.10/kWh.

 doi:10.20944/preprints202602.1854.v1

Integration effort.

Approximately 200 lines of Python overriding `AutoDeviceMapping` are required; one-time FPGA bitstream compilation takes ≈4 hours on Vitis HLS. Migration from an existing ROLL deployment: ≈1–2 person-days.

Multi-FPGA TP for larger RMs.

Two-way tensor parallelism across two U55Cs extends the RM size limit to ≈26 B with only ≈8 % PCIe synchronisation overhead.

5.2. Generalisability

EcoRL-Sched is algorithm-agnostic: GRPO (25–35 % energy reduction), REINFORCE++, and DAPO all benefit consistently (Figure 8D). For PRMs [17], more frequent per-step inference calls further increase FPGA utilisation. The scheduling algorithm is portable to any high-tok/J inference accelerator (Groq LPU, Cerebras CS-3, Alveo V80).

5.3. Lifecycle Carbon Analysis

Manufacturing 4 Alveo U55Cs: 180–280 $kgCO_2e$ embodied. Amortised over 50 training runs: 3.6–5.6 $kgCO_2e$/run. This is <3 % of the 138–1,859 $kgCO_2e$ operational savings, yielding a net carbon benefit >30× under conservative assumptions.

Embodied carbon provenance.

The 180–280 $kgCO_2e$ range is derived from two independent sources: (1) the ACT framework [39], which provides a per-mm^2 estimate for 7 nm logic (≈0.4 $kgCO_2e/cm^2$ fab), with the U55C die ≈116 mm^2 ×4 dies, giving ≈185 $kgCO_2e$; and (2) a proportional scaling from the published 16 nm Kintex UltraScale+ carbon disclosure (≈38 kg/device [47]), scaled by the ratio of transistor counts and process node emission factors per [48]. The boundary follows the ACT methodology [39], including IC fabrication energy, packaging, and PCB mounting, but excluding FPGA end-of-life/recycling (<5 % of manufacturing), server chassis, cooling infrastructure, and network switches.

5.4. Limitations

RM size is currently limited to ≈7–13 B by U55C HBM (16 GB INT8); multi-FPGA TP extends this to ≈26 B; larger RMs require next-gen platforms (Alveo V80: 32 GB HBM3, PCIe 5.0). The offline scheduler policy may need periodic online adaptation under large workload distribution shifts.

The following results are *physically measured* on our 32×H100 + 4×U55C cluster: end-to-end throughput and energy efficiency (all model sizes 7B–70B), FPGA resource utilisation (Table 2), quantisation error bound (over 160k HH-RLHF pairs), and strong scaling to 128 GPUs. The following are *model-based projections*: strong scaling 256–1,024 GPUs (±5 pp uncertainty), 405B energy numbers (projected via per-token cost scaling from 70B measurements), and multi-FPGA TP scaling to 26B RM.

6. Conclusions

We presented EcoRL-Sched, an energy-aware heterogeneous GPU–FPGA task scheduling framework for sustainable RLHF training. By characterising the intrinsic power asymmetry across RLHF stages, offloading frozen-weight inference to FPGAs at 4.9× better tokens/Joule, and learning dynamic scheduling via PPO on ROLL workloads, EcoRL-Sched achieves up to 14.6× throughput speedup, 38.4 % energy reduction, and 40.6 % CO_2 reduction vs. DeepSpeed-Chat, strictly dominating all prior systems simultaneously on throughput, energy, and carbon, without sacrificing model quality. Lifecycle analysis confirms savings exceed embodied FPGA overhead by >30×.

The core architectural insight is that the RLHF pipeline's heterogeneity is an asset: matching frozen small-batch inference to FPGAs and large-batch generation/training to GPUs, then scheduling the result intelligently with learned policies, yields substantial gains in both efficiency and sustainability.

 doi:10.20944/preprints202602.1854.v1

Future directions include multi-FPGA tensor-parallel RM offloading, process reward model integration, neuromorphic offloading for reward scoring, and carbon-aware multi-site scheduling.

Author Contributions: Conceptualization, S.E.; methodology, S.E.; software, S.E.; validation, S.E.; formal analysis, S.E.; investigation, S.E.; resources, S.E.; data curation, S.E.; writing, original draft preparation, S.E.; writing, review and editing, S.E.; visualization, S.E.; project administration, S.E. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Code, scheduler weights, FPGA bitstreams, and configurations will be made publicly available at https://github.com/Saher-Elsayed/EcoRL-Sched upon acceptance.

Acknowledgments: The author thanks the Alibaba ROLL team and the open-source communities behind vLLM, DeepSpeed, and OpenRLHF.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RLHF	Reinforcement Learning from Human Feedback
LLM	Large Language Model
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
PPO	Proximal Policy Optimisation
GRPO	Group Relative Policy Optimisation
EDI	Energy Density Index
RM	Reward Model
PRM	Process Reward Model
DMA	Direct Memory Access
HBM	High Bandwidth Memory
PCIe	Peripheral Component Interconnect Express
MLP	Multi-Layer Perceptron
HLS	High-Level Synthesis
MHA	Multi-Head Attention
GEMM	General Matrix Multiply
SFT	Supervised Fine-Tuning
ICT	Information and Communications Technology
PUE	Power Usage Effectiveness
ROI	Return on Investment

Appendix A. FPGA Synthesis Configuration

This appendix documents the complete FPGA synthesis directives and Vitis HLS pragmas used to achieve the resource utilisation reported in Table 2.

Appendix A.1. HLS Pragmas

The following Vitis HLS directives were applied to the INT8 GEMM and MHA cores: `HLS PIPELINE II=1` on all inner loops; `HLS ARRAY_PARTITION` cyclic factor 16 on weight buffers; `HLS DATAFLOW` at the top-level function boundary; `HLS INTERFACE` mode = `m_axi` on HBM ports with burst length 256. Full synthesis script, constraint files, and post-place-and-route timing reports are included in the supplementary code repository.

 doi:10.20944/preprints202602.1854.v1

Appendix B. Hyperparameter Configuration

Table A1. Complete hyperparameter configuration for all experiments.

Parameter	Value	Notes
Learning rate (actor)	1×10^{-6}	AdamW, $\beta = (0.9, 0.999)$
Learning rate (critic)	5×10^{-6}	AdamW, $\beta = (0.9, 0.999)$
PPO clip ε	0.2	Standard
KL coefficient β	0.05	Adaptive in sensitivity runs
Rollout batch size	256	Per GPU
Mini-batch size	32	Per update
PPO epochs per iter	1	Single-pass
Scheduler MLP layers	256–128–64	ReLU + LayerNorm
Scheduler LR	3×10^{-4}	Adam
Scheduler training steps	2,000	50 episodes
EDI smoothing α	0.05	EMA
Power sampling (GPU)	100 ms	NVML
Power sampling (FPGA)	50 ms	XRT

References

1. Christiano, P.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; Amodei, D. Deep reinforcement learning from human preferences. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
2. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
3. Anthropic. Claude: A Constitutional AI Assistant. Technical Report, 2023.
4. Meta AI. The Llama 3 Herd of Models. *arXiv* **2024**, arXiv:2407.21783.
5. Qwen Team. Qwen2.5 Technical Report. *arXiv* **2024**, arXiv:2412.15115.
6. DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv* **2025**, arXiv:2501.12948.
7. Freitag, C.; Berber, M.; Kleiner, A. The Climate Impact of ICT: A Review of Estimates, Trends and Regulations. *arXiv* **2021**, arXiv:2102.02622.
8. Masanet, E.; Shehabi, A.; Lei, N.; Smith, S.; Koomey, J. Recalibrating global data center energy-use estimates. *Science* **2020**, *367*, 984–986.
9. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
10. Alibaba Group. ROLL: Reinforcement Learning Online Serving Library. *GitHub* **2025**. Available online: https://github.com/alibaba/roll (accessed on 1 January 2025).
11. DeepSeek-AI. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv* **2024**, arXiv:2402.03300.
12. Hu, J.; et al. REINFORCE++: An Efficient RLHF Algorithm with Robustness and Simplicity. *arXiv* **2025**, arXiv:2501.03262.
13. Yu, Q.; et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv* **2025**, arXiv:2503.14476.
14. Bradley, R.A.; Terry, M.E. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* **1952**, *39*, 324–345.
15. Jiang, Y.; et al. PairRM: Heuristic Rejection Sampling with Preference Ranking. *arXiv* **2024**, arXiv:2306.02561.
16. Skywork Team. Skywork-Reward: Bag of Tricks for Reward Modeling for LLMs. *arXiv* **2024**, arXiv:2410.18451.
17. Lightman, H.; et al. Let's Verify Step by Step. *arXiv* **2023**, arXiv:2305.20050.
18. Yao, Z.; et al. DeepSpeed-Chat: Easy, Fast and Affordable RLHF Training of ChatGPT-like Models at All Scales. *arXiv* **2023**, arXiv:2308.01320.
19. Hu, J.; et al. OpenRLHF: An Easy-to-Use, Scalable and High-Performance RLHF Framework. *arXiv* **2024**, arXiv:2405.11143.
20. Sheng, G.; et al. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv* **2024**, arXiv:2409.19256.
21. Zhong, Y.; et al. RLHFuse: Efficient RLHF Training for Large Language Models with Inter- and Intra-Stage Fusion. *arXiv* **2024**, arXiv:2409.13221.

 doi:10.20944/preprints202602.1854.v1

22. Wu, Y.; et al. LlamaRL: An Efficient and Scalable Distributed RL Training Framework. *arXiv* **2025**, arXiv:2504.08436.
23. StreamRL Team. StreamRL: Scalable, Heterogeneous, and Elastic Scheduling for LLM Reinforcement Learning. *arXiv* **2025**, arXiv:2504.15930.
24. HetRL Team. HetRL: Heterogeneous GPU Cluster Training for RLHF. *arXiv* **2025**, arXiv:2504.13030.
25. Zeng, S.; et al. FlightLLM: Efficient Large Language Model Inference with a Complete Mapping Flow on FPGAs. In *Proceedings of the ACM/SIGDA FPGA*; ACM: New York, NY, USA, 2024.
26. Gu, C.; et al. LLM-FPGA: Efficient LLM Inference Using FPGAs. In *Proceedings of ICCAD*; IEEE: Piscataway, NJ, USA, 2024.
27. Chen, Y.; et al. A Survey on FPGA Acceleration for Transformer Models. *arXiv* **2023**, arXiv:2306.15547.
28. Xu, C.; et al. Survey of Hardware Accelerators for Large Language Models. *arXiv* **2023**, arXiv:2401.09890.
29. Etinski, M.; Corbalan, J.; Labarta, J.; Valero, M. Understanding the future of energy-performance trade-off via DVFS in HPC environments. *J. Parallel Distrib. Comput.* **2012**, *72*, 579–590.
30. Kaur, T.; Chana, I. Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM Comput. Surv.* **2014**, *47*, 1–45.
31. Pagani, S.; et al. Energy efficiency on MPSoCs with power domains. *IEEE Trans. Comput.* **2015**, *65*, 1–14.
32. You, J.; et al. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *Proceedings of NSDI'23*; USENIX: Berkeley, CA, USA, 2023.
33. Sia Team. Sia: Heterogeneity-Aware, Goodput-Optimized ML-Cluster Scheduling. In *Proceedings of SOSP'23*; ACM: New York, NY, USA, 2023.
34. Qiao, A.; et al. Pollux: Co-Adaptive Cluster Scheduling for Goodput-Optimized Deep Learning. In *Proceedings of OSDI'21*; USENIX: Berkeley, CA, USA, 2021.
35. Wiesner, D.; et al. Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. In *Proceedings of Middleware'21*; ACM: New York, NY, USA, 2021.
36. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *Commun. ACM* **2020**, *63*, 54–63.
37. Lacoste, A.; et al. Quantifying the Carbon Emissions of Machine Learning. *arXiv* **2019**, arXiv:1910.09700.
38. Courty, B.; et al. CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing. *J. Open Source Softw.* **2023**, *8*, 4403.
39. Gupta, U.; et al. ACT: Designing Sustainable Computer Systems with an Architectural Carbon Modeling Tool. In *Proceedings of ISCA'22*; ACM: New York, NY, USA, 2022.
40. Patterson, D.; et al. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer* **2022**, *55*, 18–28.
41. Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *arXiv* **2022**, arXiv:2208.07339.
42. Bai, Y.; et al. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv* **2022**, arXiv:2204.05862.
43. NVIDIA. NVML API Reference Manual. Available online: https://docs.nvidia.com/deploy/nvml-api/ (accessed on 1 January 2025).
44. Shah, J.; et al. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-Precision. *arXiv* **2024**, arXiv:2407.08608.
45. Rajbhandari, S.; et al. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. In *Proceedings of ICML'22*; PMLR: Cambridge, MA, USA, 2022.
46. International Energy Agency. Electricity 2023: Analysis and Forecast to 2025. Available online: https://iea.org/reports/electricity-2023 (accessed on 1 January 2024).
47. Xilinx. Sustainability Report 2021. Available online: https://www.amd.com/en/corporate/sustainability (accessed on 1 January 2022).
48. Hintemann, R.; Hinterholzer, S. Energy Consumption of Data Centers Worldwide. *Borderstep Institute* **2022**. Available online: https://www.borderstep.de (accessed on 1 January 2023).