

Stateful Runtime Adaptation under Workload and Thermal Change: An Audited Evaluation on a Heterogeneous SoC

Antonio Souto Rodriguez

`antonio.rodriguez@icomp.ufam.edu.br`

Federal University of Amazonas

José Reginaldo Hughes Carvalho

Federal University of Amazonas

Research Article

Keywords: runtime resource management; heterogeneous systems; evolutionary control; surrogate learning; thermal management; experimental evaluation

Posted Date: October 28th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11516599/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Stateful Runtime Adaptation under Workload and Thermal Change: An Audited Evaluation on a Heterogeneous SoC

Antonio Souto Rodriguez*
José Reginaldo Hughes Carvalho

Institute of Computing, Federal University of Amazonas, Manaus, Brazil

Abstract

We evaluate an evolutionary runtime controller on a Jetson Nano using six six-hour dynamic missions and twelve static-policy missions. Four vision workloads recur three times per mission. The evaluation distinguishes requested actions, software readback, runtime outcomes and retained controller state. All 66 dynamic workload switches preserve the logged generation counter. Under the aggressive thermal profile, Median filtering has 19.8% higher reported throughput and 14.4% lower sampled power than the GPU-static policy, but the CPU-static policy remains faster. Under the inactive profile, controller throughput is 19.0% and 22.0% below the best observed static centres for Gaussian and Median workloads. Forecast error does not consistently decline on recurrence. Two repetitions per condition, unverified output equivalence, unmatched input and thermal histories, and incomplete source-to-binary provenance limit causal interpretation. The results establish persistent operation and heterogeneous trade-offs rather than general optimization superiority.

Keywords: runtime resource management; heterogeneous systems; evolutionary control; surrogate learning; thermal management; experimental evaluation

1 Introduction

An embedded vision system must coordinate acquisition, processing, display and resource management on a shared power and thermal budget. Moving a kernel to a GPU changes only part of that system: capture conversion, synchronization, transfers and display can continue to load the CPU. A runtime controller consequently chooses a policy for a concurrent pipeline rather than simply choosing the fastest isolated kernel. Workload changes add a state-management problem. Resetting the optimizer or its prediction model at every change discards information, while retaining state exposes later decisions to earlier observations and to any attribution errors in those observations.

This paper examines the implemented Benchmark-Integrated ERL framework, denoted ERL. Its final learning component is a workload-conditioned empirical surrogate coupled to persistent evolutionary search. Adaptive objective weights and an application-level thermal governor mediate resource decisions before a scheduler publishes controls. The study concerns this final architecture; it does not evaluate a separate reinforcement-learning policy. Pareto ranking, surrogate assistance and adaptive scalarization are established ingredients[1, 2]; the contribution here is their integration with continuous workload replacement and an explicit experimental assessment of the resulting runtime behavior.

*Corresponding author: antonio.rodriiguez@icomp.ufam.edu.br.

A central evaluation difficulty is that a requested resource setting is not necessarily the setting applied by the operating system or used by a worker. Similarly, a valid event checksum does not establish a fresh sensor observation, and retained model entries do not establish a learning benefit. These distinctions motivate a recording-based evaluation that checks the relationships among configuration, readback, observation and claim. Performance measurements also depend on system state and experimental units[3, 4]. We therefore retain every supplied repetition and report both favorable and unfavorable outcomes.

The campaign contains six dynamic and twelve static-policy missions, each lasting approximately six hours. Four image workloads recur three times within a mission. Dynamic missions cover inactive, moderate and aggressive application thermal profiles; static policies cover the inactive and aggressive profiles. The policies share a nominal workload schedule and WARNING GPU-withdrawal predicate. They do not constitute randomized, input-identical or thermally identical trials. The resulting comparisons characterize the recorded implementations under their observed operating conditions.

We address four questions. **RQ1:** Does the controller maintain observable state and progress across workload replacement, and what computation cost is recorded? **RQ2:** What throughput, processing-latency, power and temperature trade-offs occur relative to schedule-matched static policies? **RQ3:** What evidence supports workload-memory retention, forecast usefulness and the interpretation of individual selection failures? **RQ4:** Which conclusions survive the actuation, timing, provenance and comparability checks?

The paper contributes (i) an explicit account of persistent search, workload-indexed prediction, adaptive weighting and runtime mediation; (ii) a replicated, 108-hour descriptive comparison with traceable channel definitions and separate run-level estimands; and (iii) an analysis of where apparent optimization claims fail, including retained-state versus recurrence benefit, GPU permission versus executed work, and composite scores versus their underlying measurements. It does not claim a new general-purpose optimization algorithm, global optimality or a causal advantage from an individual controller mechanism.

2 Related work

Runtime adaptation on heterogeneous systems spans several decision layers. Tangram coordinates subsystem controllers through a hierarchy that combines optimization and safety mechanisms, and evaluates this design on a heterogeneous server [5]. This work establishes that policy composition and separation of optimization from enforcement are existing systems concerns. ERL studies their interaction in a smaller setting: a persistent evolutionary search, workload-conditioned prediction, adaptive objective weights, a thermal supervisor, and runtime controls on one embedded CPU/GPU platform. This integration should be assessed through its specified interfaces and observed behavior, without claiming that the individual mechanisms are new.

Learning-based schedulers address related but different decisions. DeepSoCS uses hierarchical representations of jobs and tasks to learn scheduling decisions, with its evaluation conducted through simulation [6]. SoCRATES focuses on steady-state task latency and the concurrency of observations when dependent tasks execute on heterogeneous processing elements [7]. Cohmeleon instead learns runtime coherence-mode selections for accelerators and evaluates these decisions using FPGA prototypes [8]. These systems differ in actions, workloads, platforms, and measured outcomes; their reported improvements are therefore not numerical baselines for this study. ERL’s evaluated mechanism is persistent evolutionary/Pareto search aided by workload-conditioned surrogate observations. Its name does not establish that the implementation contains a separate reinforcement-learning policy.

The algorithmic components also have established foundations. NSGA-II combines nondominated ranking, crowding-based diversity, and elitist environmental selection [1]. Employing

ranking and crowding alone does not justify describing a controller as an unmodified implementation of that complete algorithm; the reproduction and population-update rules must be specified. Surrogate-assisted evolutionary computation uses models to reduce costly fitness evaluations, with model management needed to keep approximate evaluations grounded in observations [2]. Accordingly, retaining workload-indexed observations is distinct from demonstrating that retention improves prediction or control. Such a benefit requires comparisons that separate memory from changes in workload, temperature, and enforced action.

Experimental validity is central when a controller both acts on a system and reports its own behavior. Hoefler and Belli provide guidance on experimental design, uncertainty, and reproducible performance reporting [3]. Mytkowicz et al. show that seemingly minor setup differences can bias systems experiments [4]. These findings motivate explicit treatment of replicate identity, arm comparability, and measurement boundaries here. They do not make dense samples from one mission independent experimental replicates. The measurement interpretation also follows platform semantics: Linux documents that `scaling_cur_freq` may report a requested performance state rather than the instantaneous physical clock, while NVIDIA describes interactions between CPU-frequency control and thermal management on Jetson Nano [9, 10]. Consequently, requested settings, policy readback, enabled actions, and measured outcomes must remain separate quantities.

The contribution examined in this paper is the experimentally bounded behavior of the integrated controller during recurrent workloads and imposed thermal profiles, compared with fixed operating modes under explicitly qualified measurement boundaries. Static comparisons can describe the aggregate trade-offs of the integrated implementation. They do not, by themselves, establish a causal benefit from memory, adaptive weights, or evolutionary search individually. No claim of universal superiority over the cited systems follows from this evaluation.

3 Controller and experimental system

3.1 Persistent controller architecture

ERL combines a persistent evolutionary population, workload-conditioned surrogate estimates and an adaptive weight manager (AWM) with a thermal-mediated scheduler. The execution plane comprises camera acquisition, a replaceable image-processing worker, display and telemetry. The selector owns the genetic algorithm and scheduler; the genetic algorithm owns the surrogate and AWM. This organization preserves controller identity while workloads change. The learning component in this implementation is the empirical surrogate. The architecture does not require a separate policy/value-network reinforcement-learning agent.

The nominal controller interval is 800 ms. Population exports contain eight evaluated candidates per recorded generation, independently corroborating the configured population size. The supplied source describes the controller mechanics below, whereas run records establish observed settings and outcomes. Their provenance is kept separate: the manifests declare binary/configuration identities but leave source hashes unset, so the supplied source cannot be asserted to be the exact compiled revision of every recording.

3.2 Search, estimation and deployment

The genome is $u = (m, r, b, g, f, s)$: policy mode, target frame rate, power budget, GPU preference, CPU-frequency cap and GPU work fraction. Worker count and affinity are derived by mode-dependent mappings and subsequent scheduling; they are not independent genome coordinates. The recorded CPU-frequency gene levels are 102, 403.2, 921.6, 1132.8 and 1479 MHz. A workload capability rule restricts split to binary values except for heterogeneous Gaussian processing. The

requested gene, model-query configuration, written cap, readback and worker settings are distinct quantities.

Candidate evaluation minimizes four predicted objectives. Power is normalized watts, despite the legacy `obj_energy` name; temperature is a CPU-temperature exceedance penalty; the latency objective uses the controller’s pipeline-latency channel. The frame-rate objective generally rewards proximity to the candidate target, while the high-performance AWM regime rewards normalized throughput. Constraint-aware nondominated sorting and crowding order parents and elites. Deployment selects the highest scalar-fitness feasible rank-zero candidate, with feasible-candidate and least-violation fallbacks. Elite retention and tournament/crossover/mutation then form the next generation. These are Pareto-based evolutionary operators, not full parent-plus-offspring NSGA-II environmental selection or a retained archive of all historical fronts.

The surrogate projects a genome into (f, g, s, c) and selects a bank by workload identity. Bucket keys quantize frequency to the nearest gene level, split to quarters and concurrency to an integer in $[1, 4]$. A populated bucket returns exponential moving averages (EMAs); otherwise the model interpolates up to three neighbours with weights $1/(10^{-3} + d)$, then falls back to a specified prior when no observations exist. Here d is Euclidean distance over normalized features: clamped frequency fraction, binary GPU preference, canonical split and $(c - 1)/3$. The neighbour query retains continuous canonical split and the source frequency rather than replacing every feature with its bucket key. Initial valid-target updates use $\alpha = 0.60$, subsequent updates $\alpha = 0.30$. A count-based gate limits low-support throughput predictions, and an inverse-square-root visit bonus encourages exploration. Counts are shared across targets, so support does not certify that each target was measured; confidence is heuristic rather than calibrated uncertainty.

AWM samples operating conditions every five generations, commits a regime after two qualifying detections and smooths weights with $\alpha = 0.30$. Its priority is critical temperature, critical power, warning temperature, latency exceedance, underperformance and nominal operation. These weights express soft preferences. The downstream ThermalGovernor evaluates $\max(T_{\text{CPU}}, T_{\text{GPU}})$ against profile thresholds; the scheduler can withdraw GPU permission and transform the request. This application-level mediation is distinct from vendor thermal protection and does not establish a hard temperature or real-time guarantee.

One normal control period.

1. Read the latest snapshot; require useful values, a new frame identifier and the active workload epoch. Collector hooks close the preceding ticket and open the next context.
2. Enter the GA stability gate; update AWM when due, then update the active workload model from pending previous-request credit, if present.
3. Predict and gate candidate outcomes; calculate constraints, objectives and scalar fitness; rank and select a candidate.
4. Freeze the selected evaluation; apply the request through the scheduler/governor and record commanded state and available readback.
5. Export evaluated candidates, retain elites and generate offspring; advance the generation and publish selected-action events.

The next accepted snapshot is associated with the preceding request. Frame/workload freshness and a ticket join do not prove that all constituent sensor samples belong to a settled effective action. Recovery overrides, model/request mapping and workload-boundary ticket handling therefore constrain causal interpretation of predictions.

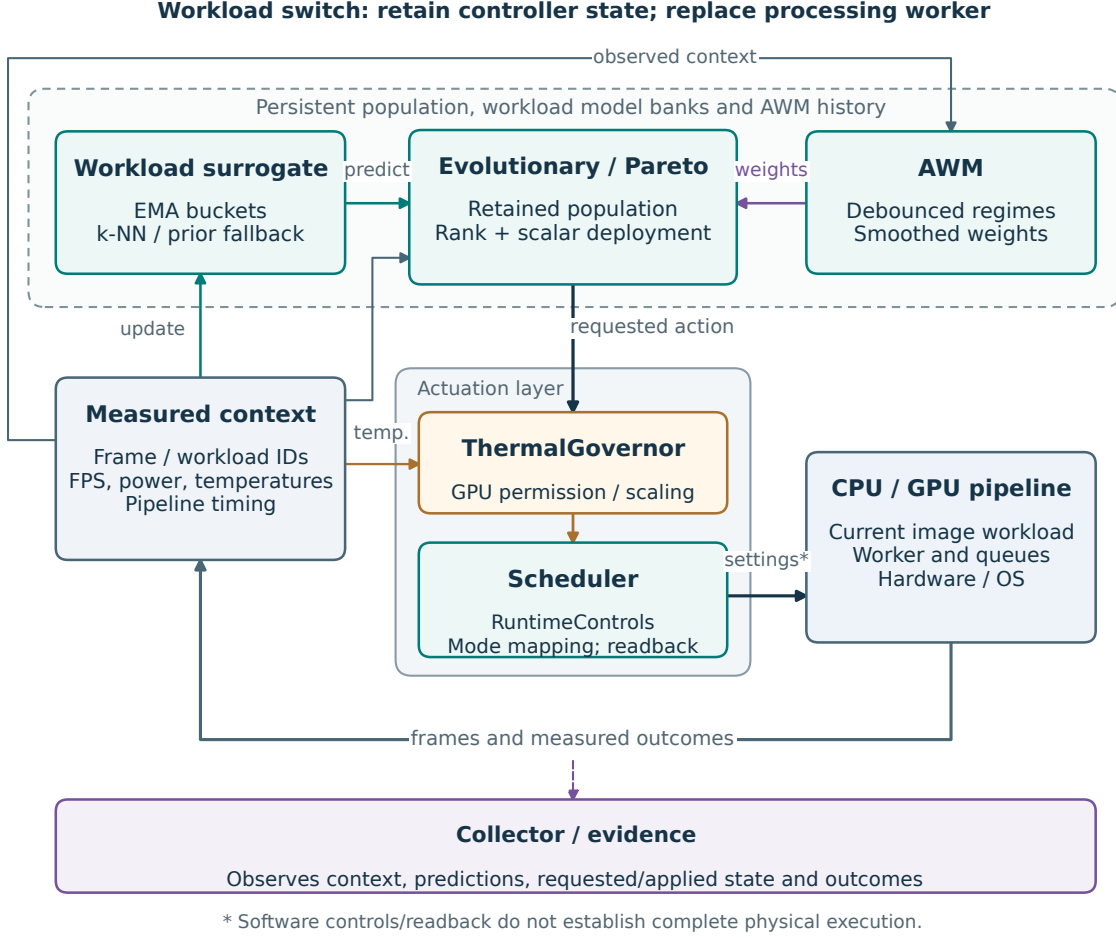


Figure 1: Reference architecture and its control/observation paths. Persistent search, workload-conditioned prediction and AWM generate requests; the governor and scheduler mediate deployment. Collected software state and outcomes remain separate evidence. The source snapshot is not cryptographically bound to the historical binaries

3.3 Workload switching

At a boundary, a transition flag and mutex barrier wait for the current evolutionary period. A replacement algorithm is prepared, the old worker is stopped/joined, processing queues are drained and a new worker epoch is activated. GA context changes in place, selects the retained workload bank, canonicalizes incompatible genes and clears pending GA learning credit. Generation, population lineage, model banks, AWM history and scheduler identity persist; the population is not necessarily byte-identical. Fresh-frame/epoch checks and the caller’s transition evidence document continuity. Retention itself does not demonstrate faster recovery or beneficial transfer.

4 Evaluation method

4.1 Campaign and provenance

Each mission follows HistogramEqualization, SobelEdge, HeterogeneousGaussianBlur and MedianFilter, in that order, for three visits of 1,800s per workload. This gives twelve phases and eleven replacements per mission. The six ERL missions use two chronological repetitions of each thermal profile. The twelve static missions use CPU_STATIC, GPU_STATIC and

BALANCED, with two repetitions of each under inactive and aggressive conditions. There is no moderate-profile static cell. The caution/warning/critical thresholds are 75/85/95 °C (inactive), 55/62/70 °C (moderate) and 45/50/55 °C (aggressive). Profile names denote configured policy thresholds, not guaranteed temperatures. CPU_STATIC requests four CPU workers with GPU disabled; GPU_STATIC requests one worker and full GPU split; BALANCED requests two workers and half split. The static launcher omits GA, AWM and Scheduler and applies its own strict WARNING-stage GPU withdrawal rule. Online Resource 1 lists all 18 mission identifiers and record counts.

The recorded platform is a Jetson Nano 2 GB. Static startup records specify a CSI RG10 input of 1,280×720 at a requested 60 frames/s, converted to 320×240 YUYV. These are configuration values; the realized capture-loop rate differs between runs, and identical live scene content is not established. The ERL and static manifests declare different executable hashes, beginning d7235178 and 9d05f40d, respectively. They share the declared configuration hash beginning 7e3ecb06. Source hashes and build flags are unset, so the difference cannot be certified as launcher-only. The inspected reference source is described separately from as-run observations.

Repetition numbers identify chronological recordings. In particular, all six dynamic manifests carry the same block identifier, so matching that field to static identifiers would falsely imply paired experimental blocks. Runs were recorded on different days. Ambient conditions, starting temperature, input content and full external platform state were not controlled sufficiently to support a randomized causal comparison. Both repetitions, including the unusually fast aggressive CPU_STATIC recording, remain in the primary results.

4.2 Admission by measurement channel

We avoid a single blanket “valid run” label. Schedule and transition checks establish recorded phase order. Frame and action identifiers support joins. Event CRCs and final digests establish file integrity within the audited records. Separate predicates address the meaning and consistency of individual measurements.

CPU policy state. C1a compares the written/commanded cap with software policy readback. C1b compares the reported current-state value with the written cap. C1a does not validate physical clock residency or unconditional fidelity to the original genome. We retain C1b exceptions as observed behavior and supply a cap-consistent sensitivity analysis; no result is attributed to physical-frequency compliance.

Latency and throughput. The primary latency outcome is the recorded processing interval, excluding input-queue waiting and downstream presentation. Legacy end-to-end or composite channels are not relabeled as sensor-to-output latency. FPS is the sampled reported throughput estimator; it is not an independent count of numerically equivalent delivered outputs. A zero recorded latency-invariant violation or sampled starvation count does not establish lossless delivery. Console drop and pruning notices are retained in the evidence record.

GPU state and thermal mediation. Enabled flags and split values describe permission/configuration. They do not, alone, establish per-frame GPU execution. The dynamic denial audit is conditional on a GPU request and uses the logged predicate $\max(T_{\text{CPU}}, T_{\text{GPU}}) > \theta_{\text{warning}}$. Equality is on the permitted side of the predicate. Static arms implement the WARNING gate, not the entire dynamic supervisor/AWM path. Gate adherence is distinct from maintaining a temperature below the threshold.

Power and energy. Whole-mission energy integrates channel-0 voltage times current on host timestamps, restricted to the manifest window. Intervals longer than three seconds are removed in a supported-interval sensitivity. Workload power summaries use the reported trace power channel; these are distinct estimands from integrated energy divided by mission duration. We do not divide median power by median FPS and call the result energy per output. Rail

attribution, sensor freshness and equivalent useful output require additional evidence.

Output and implementation equivalence. Per-frame CPU/GPU output hashes or numerical comparisons on identical input are absent. The reference source also contains different Sobel operators, a constant-image Histogram difference and a partial-Gaussian halo problem. Its identity with the recorded executables is unverified. Accordingly, cross-policy timings are implementation-path observations, not demonstrated speedups for equivalent numerical work. Median has compatible CPU/GPU algorithmic intent in the inspected source, but device-level equivalence remains unmeasured.

4.3 Estimands and uncertainty

For policy a , profile p , run r and workload w , the primary run summary is

$$m_{a,p,r,w,k} = \text{median}\{x_{t,k} : t \text{ belongs to workload } w \text{ in run } r\}, \quad (1)$$

where the three visits are pooled within a run and k identifies a measured channel. A cell centre is the arithmetic mean of the two run medians,

$$\bar{m}_{a,p,w,k} = \frac{1}{2}(m_{a,p,1,w,k} + m_{a,p,2,w,k}). \quad (2)$$

Samples are not pooled across repetitions. Figures show each run and their observed range, not a population confidence interval. Relative comparisons use $100(\bar{m}_{\text{ERL}}/\bar{m}_a - 1)$. The best observed static throughput centre is selected from the three supplied modes; it is not an oracle over all possible resource configurations.

There are two independent missions per condition. Frames, candidate rows and phases are repeated observations within those missions, not additional independent experiments. We do not report frame-level significance tests or infer equivalence from overlapping within-run intervals. Supplementary phase summaries retain 1,000 bootstrap resamples using nonoverlapping 120-s blocks with fixed seed 20261023. Those intervals describe within-run variability; eleven of 72 dynamic phases retain long dependence or trends at the 120-s diagnostic cap. Full-phase results are primary, with removal of the first 30 seconds and median-of-phase-medians reported as sensitivities.

4.4 State, computation and forecast diagnostics

Generation values immediately before and after a replacement test counter continuity. Workload-specific observation counts at returns test retained entries. These checks do not prove byte-identical population state or an accuracy benefit. Algorithm downtime is measured separately from the later first-frame proof. GA computation is summarized both as a per-period duration divided by the nominal 800-ms interval and as accumulated named GA compute time divided by elapsed mission time. Neither quantity measures the complete controller, telemetry, scheduling and operating-system overhead.

Forecasts are joined to outcomes by collector query identity, retaining the frozen pre-application prediction values. The recorded interval remains assigned rather than independently established from sensor acquisition times. In the inspected source, the issue timestamp is derived from the selection boundary (`selectNs-1`), not independently instrumented at prediction completion; if selection time is missing, the fallback derives it from application completion. Historical source-to-binary identity remains unresolved. Baseline chronology is therefore conditional on the recorded timestamps. The collector’s admission event is an observation label emitted before the model update, not proof that an observation entered the model. The reference implementation also leaves the previous collector ticket active at workload changes although GA learning credit is

cleared. We therefore exclude pairs crossing the recorded phase boundary and other phase-time inconsistencies before reporting revised diagnostics.

For an exploratory forecast comparison, persistence predicts the last available outcome of the same workload, and an exponential moving average (EMA) uses the same strictly prior outcomes with fixed $\alpha = 0.30$. Excluded boundary observations are also removed from baseline history. Comparisons use identical non-cold-start pairs and no parameter tuning. The baseline analysis is post hoc and evaluates selected trajectories; it does not establish counterfactual candidate accuracy, improved decisions or a causal memory-retention effect. Processing-latency prediction is excluded because the historical prediction target differs from processing latency.

5 Results

5.1 Coverage and software-state consistency

The campaign contains approximately 108 hours, 216 phases and 198 workload replacements: 36 hours and 66 replacements for ERL, and 72 hours and 132 replacements for static policies. The distinct record counts are 766,250 sampled trace rows, 161,723 dynamic actions and 8,077,130 retained frame-metric rows. These overlapping streams are not summed as independent control periods or complete outputs.

C1a holds in all 161,723 dynamic action records. C1b has 3,701 exceptions (2.288%), confined to the reported mappings 460,800 to 518,400 kHz and 1,035,299 to 1,036,800 kHz. Written caps differ from original requests in 8,336 records. Thus the campaign verifies software cap readback. It establishes neither downward-snapping of every requested cap nor physical-frequency conformance. Static cap commands are zero and expected/readback fields are -1 ; a cap-write equality test is inapplicable to those arms.

All 18 audited event seals and CRC checks pass. One dynamic append-order timestamp inversion is 0.270 ms; within-query event order remains valid in the preserved audit. Reference-source fault tests show why seals must remain an integrity check rather than a scientific-validity certificate. We report zero invariant violations only for the retained metric records and do not infer complete frame delivery.

5.2 State continuity and computation cost

All 66 dynamic replacements preserve the logged generation counter; all 198 replacements increment the workload epoch correctly. Returned observation counts range from 2,216 to 2,248 at visit 2 and 4,422 to 4,496 at visit 3. These are workload-memory counts, distinct from the support count of an individual selected prediction. Static arms also perform workload replacement, so replacement continuity alone is not evidence of an evolutionary advantage.

Dynamic downtime medians range from 10.9 to 24.7 ms across runs, with a maximum of 662.2 ms. No sampled starvation is recorded. Queue-drain and console drop evidence preclude a lossless-output claim. The median ratio of named GA compute time to the nominal period is 0.26–0.41%; accumulated GA compute duty is 0.752–3.479% of mission time. Seven compute records exceed 800 ms, including two in aggressive repetition 2. Table 1 reports per-run values without turning the observed maxima into a hard real-time bound.

5.3 Raw throughput and processing trade-offs

Figure 2 displays every run-level throughput summary. Table 2 compares the mean of the two dynamic run medians with the best observed static centre. ERL is below that centre in all eight available profile/workload cells. Under inactive conditions, the differences are small for

Table 1: Dynamic software-state checks and recorded computation/switch timing. Swap and GA columns are medians; P95 is GA time. Duty is accumulated named GA computation divided by mission duration. All action records pass C1a; C1b flags concern reported policy state.

Profile	Rep.	Actions	C1b flags	Swap ms	GA ms	P95 ms	GA duty (%)
Inactive	1	26907	222	14.39	2.30	26.11	0.85
Inactive	2	26972	265	13.63	2.04	22.95	0.75
Moderate	1	26976	246	11.21	2.25	23.52	0.82
Moderate	2	26972	144	10.90	2.76	25.30	0.96
Aggressive	1	26951	2621	24.72	3.24	112.48	3.48
Aggressive	2	26945	203	19.01	2.44	77.59	1.40

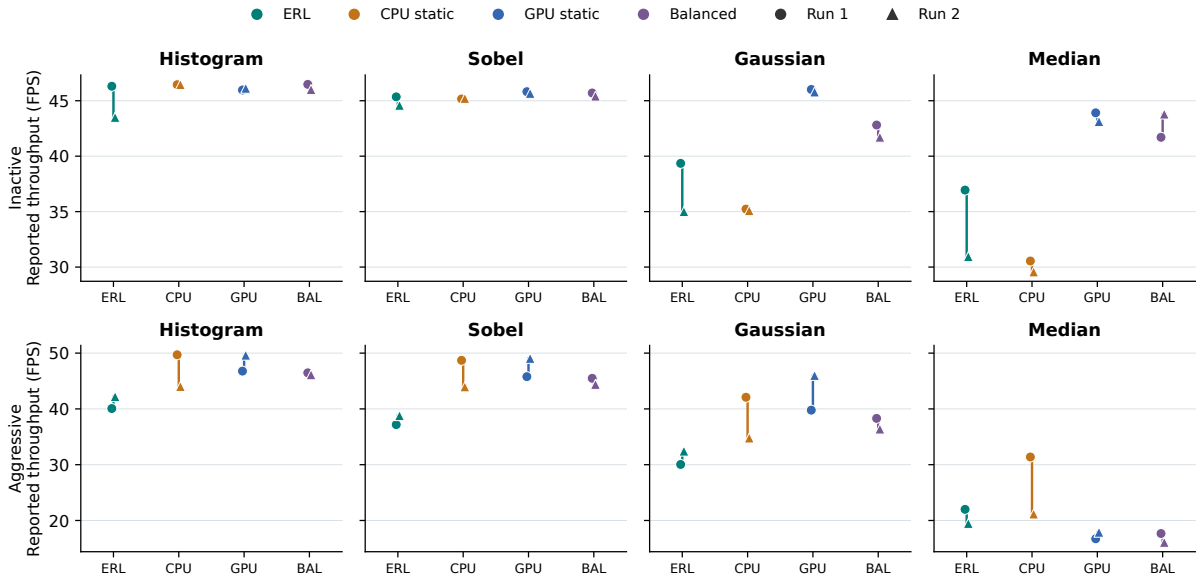


Figure 2: Run-level workload throughput under inactive and aggressive profiles. Each point is one six-hour mission’s median over three visits. Two points per policy/condition are independent mission summaries; displayed ranges are not confidence intervals. Moderate conditions have no static comparison

Histogram and Sobel (1.55 and 0.77 FPS), but reach 19.0% for Gaussian and 22.0% for Median. Gaussian processing latency also differs markedly between dynamic repetitions, approximately 6.19 versus 23.49 ms, whereas GPU_STATIC is approximately 5.58 ms. This variation is part of the result.

The favorable aggressive Median comparison is narrower than general superiority. ERL reaches a centre of 20.73 FPS, compared with 17.31 for GPU_STATIC and 16.87 for BALANCED, increases of 19.8% and 22.9%. Its processing latency is 26.1% and 26.5% lower, and sampled power is 14.4% and 16.1% lower, respectively. Both dynamic run medians exceed both corresponding GPU_STATIC and BALANCED throughput medians. CPU_STATIC is faster at a centre of 26.27 FPS, with a wider observed range. Table 3 retains all four policies. These are observed policy-bundle outcomes; they do not isolate the effect of memory, adaptive weights or a common numerical workload.

5.4 Thermal mediation and whole-mission energy

Aggressive GPU-request denials are 8,999 of 11,168 requests in repetition 1 and 7,506 of 9,941 in repetition 2. There are no request-conditional disagreements with the recorded WARNING predicate. Moderate denials are one and 40; inactive denials are zero. The resulting

Table 2: ERL throughput relative to the highest observed static centre. Entries are means of two run medians, with their observed ranges. The reference is descriptive, not an equivalent-output oracle; inactive Histogram/Sobel static separations are below the 2-FPS discrimination floor.

Profile	Workload	ERL FPS [range]	Reference	Ref. FPS [range]	Δ (%)
Inactive	Histogram	44.90 [43.50, 46.31]	CPU static	46.45 [46.45, 46.46]	-3.33
	Sobel	44.97 [44.59, 45.35]	GPU static	45.74 [45.66, 45.82]	-1.68
	Gaussian	37.18 [35.01, 39.35]	GPU static	45.90 [45.79, 46.02]	-19.01
	Median	33.95 [30.95, 36.94]	GPU static	43.52 [43.12, 43.91]	-21.99
Aggressive	Histogram	41.14 [40.07, 42.21]	GPU static	48.20 [46.77, 49.63]	-14.66
	Sobel	37.99 [37.18, 38.79]	GPU static	47.42 [45.78, 49.06]	-19.89
	Gaussian	31.24 [30.04, 32.43]	GPU static	42.89 [39.78, 46.00]	-27.16
	Median	20.73 [19.47, 21.99]	CPU static	26.27 [21.16, 31.38]	-21.09

Table 3: Aggressive-profile Median trade-offs. Entries are means of two run medians with their observed ranges; these ranges are not confidence intervals. Power is the sampled trace channel.

Policy	FPS [range]	Processing ms [range]	Power W [range]
ERL	20.73 [19.47, 21.99]	40.03 [37.08, 42.98]	5.51 [5.51, 5.52]
CPU static	26.27 [21.16, 31.38]	33.80 [26.22, 41.38]	6.50 [6.47, 6.52]
GPU static	17.31 [16.72, 17.89]	54.20 [53.12, 55.28]	6.44 [6.33, 6.55]
Balanced	16.87 [16.08, 17.66]	54.46 [50.56, 58.37]	6.57 [6.56, 6.58]

dynamic enabled-time shares are approximately 8.07% and 9.06% under aggressive conditions. GPU_STATIC enabled time is 4.80% versus 21.91%, showing materially different thermal histories even within one nominal policy/profile cell.

Inactive AWM operation is almost entirely LATENCY_SLA: 26,902 of 26,907 and 26,957 of 26,972 action records, with the remaining five and fifteen NOMINAL. The reported latency exceeds the 30-ms trigger in more than 99.7% of action periods. This establishes near-continuous trigger activation, not a physical latency floor or proof that AWM caused an outcome change. Full-run conclusions use actions rather than the 500-record tail in AWM-history exports.

Aggressive whole-mission dynamic mean power is 5.355 and 5.568 W; the static recordings range from 6.084 to 6.448 W. Dynamic energies are 115.67 and 120.27 kJ. Figure 3 shows mission-level measured channel energy separately from workload power medians. Lower energy over equal duration accompanies lower reported throughput on most workloads; it is not evidence of lower energy per equivalent output. Starting temperature, scene content and ambient conditions are not matched, and the dynamic peak temperature is not always below every static peak.

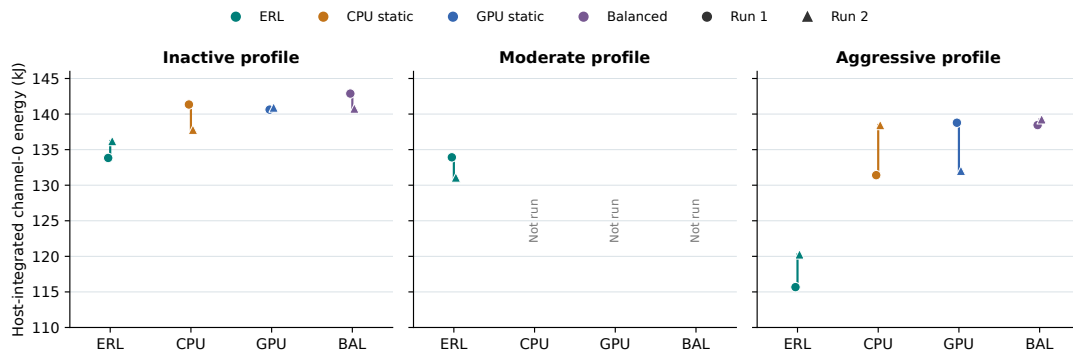


Figure 3: Host-timestamp integrated channel-0 energy over each approximately six-hour manifest window. Each point represents one mission. Equal duration permits a measured-load comparison but does not equate completed useful output

Table 4: Exploratory selected-trajectory throughput forecasts: median absolute error in FPS on identical retained target pairs. Persistence and EMA histories use only earlier retained observations. Timestamp and action-window limitations remain.

Profile	Rep.	Pairs	ERL	Persistence	EMA (0.30)
Aggressive	1	26,935	3.091	5.449	4.722
Aggressive	2	26,929	2.836	3.705	3.186
Inactive	1	26,891	3.221	3.889	3.363
Inactive	2	26,956	3.027	3.724	3.164
Moderate	1	26,958	3.806	4.759	3.944
Moderate	2	26,956	3.367	4.177	3.549

The moderate repetition-1 meter stream contains a 10,721.157-s device-clock jump, not a host acquisition outage of that duration. Its largest host gap is 8.172 s. Excluding host intervals longer than three seconds retains 99.829% temporal coverage and changes mean power from approximately 6.2000 to 6.2010 W. Reported sampling cadence is measured from host timestamps; configured polling periods are not substituted for the observed approximately 1.84–1.87-s dynamic median intervals.

5.5 Retention, forecasts and selected alternatives

The original collector projection contains 161,717 scored pairs. Sixty-six cross a recorded workload-exit boundary, one per dynamic replacement; two more have prediction/application timestamps preceding the recorded workload entry. Removing these 68 pairs leaves 161,649. The final phase has no recorded exit in this projection, so its terminal bound remains open. The exclusions remove an identified boundary ambiguity without certifying every source sample’s membership in a settled action interval.

After removal of 24 workload cold starts, the comparison uses 161,625 identical target pairs. The logged ERL forecast has lower median absolute throughput error than persistence and fixed-parameter EMA in each of the six missions (Table 4). ERL error ranges from 2.836 to 3.806 FPS; persistence ranges from 3.705 to 5.449 and EMA from 3.164 to 4.722. ERL and EMA predictions coincide within 10^{-6} in 7.8–49.0% of comparable pairs. This supports usefulness relative to these two simple predictors of the recorded selected-trajectory target, without establishing better control decisions or counterfactual configuration accuracy.

Visit-3 median absolute error is below visit 1 in only one mission, aggressive repetition 2. Reported exact-bucket source status covers 99.54–99.88% at visit 2 and 99.69–100% at visit 3. Retention and frequent exact-bucket lookup therefore coexist with no consistent recurrence-associated reduction in error. Shared support counts are not proof of independently fresh observations for every target. Requiring a prediction to occur at least 1, 5 or 10 seconds after phase entry changes each mission’s retained-set FPS MedAE by at most 0.0107 FPS.

Candidate records provide a more precise account of the weak inactive Gaussian repetition than the earlier “GPU lock-out” explanation. The six exported populations contain 1,293,784 candidate rows, eight per generation. The exported-selection audit finds one selected candidate per generation; where feasible rank-zero alternatives exist, selection agrees with their maximum exported scalar fitness within the audit tolerance. This checks exported behavior, not an independent recomputation of dominance or every internal mechanism.

In inactive Gaussian repetition 2, generations 4,681–4,682 request GPU permission at a 102-MHz CPU cap, while the logged runtime split is zero. They are not established GPU-execution trials or 102-MHz GPU-cap experiments. After generation 4,682, 6,574 GPU-preferred alternatives still appear across 2,655 generations; 2,260 generations contain a feasible rank-zero GPU alternative, and the selected CPU candidate has higher logged fitness in each. Candidate

generation therefore continued. The logs support a model/selection diagnosis but not a causal claim that retained memory alone created or sustained the underperformance.

5.6 Repetition and score sensitivity

The aggressive CPU_STATIC processing medians change from 2.51 to 10.24 ms for Sobel, 13.19 to 22.90 ms for Gaussian and 26.22 to 41.38 ms for Median between repetitions. The slower recording still reports the nominal current-state value in its sampled rows. Recorded configuration agreement does not identify the cause. CPU saturation and differences among worker-count regimes are descriptive observations, not controlled evidence of acquisition-induced contention or negative thread scaling. Neither repetition is discarded.

Removing the first 30 seconds of each phase changes static medians by at most 0.347 FPS, 0.178 ms and 0.127 W; dynamic maxima are 1.372 FPS, 1.983 ms and 0.096 W. Cap-consistent and alternative aggregation sensitivities are supplied with Online Resource 1. These diagnostics assess robustness of summaries, not restoration of otherwise unmeasured output or environmental equivalence.

The normalized reference-gap construct is retained in Online Resource 1 as a secondary, explicitly post-hoc diagnostic. The fixed four-metric weights are 0.50/0.20/0.20/0.10 for FPS/power/temperature/processing latency. With the stated discrimination floors, the strict score is defined only for aggressive Sobel; its direction changes under alternative weights. A two-metric throughput-latency variant gives inactive Gaussian 0.514/1.015 and Median 0.404/0.800 for repetitions 1/2. Dropping metrics changes the question. Neither score establishes proximity to a global optimum, and output-equivalence and reference-state limitations apply equally to normalized results. Individual measured channels lead the interpretation.

6 Discussion

6.1 What the implemented controller contributes

The strongest supported result is sustained integration: a persistent evolutionary controller continues across workload replacement while keeping workload-indexed observations and exposing requested, mediated and recorded state. The positive aggressive Median case demonstrates a useful operating trade-off relative to two supplied static policies. The weaker inactive results and inter-run variation show why that favorable case cannot be generalized to the controller as a whole. This is an empirical systems contribution with explicit operating limits, rather than evidence that standard evolutionary or surrogate techniques are intrinsically superior.

The measurement distinctions are operationally important. Logged CPU-cap equality would conceal residual reported-state exceptions if treated as physical actuation proof. An enabled GPU with zero split is not a completed accelerator trial. A retained model bank can persist without improving recurrence error. A successful event seal can coexist with an invalid measurement interpretation. These examples identify requirements for evaluating adaptive software on hardware whose acquisition, compute and control paths interact.

6.2 What would establish individual mechanism benefits

Establishing a benefit from memory retention requires an intervention that resets only the workload model, with an otherwise unchanged controller and a sham comparison. AWM requires fixed-versus-adaptive weights under the same supervisor and objective definitions. Search contribution requires a matched search-budget comparator. Turning the current surrogate flag off is not a clean isolated ablation: the inspected reference implementation also changes objective

arithmetic and bonus/penalty behavior. These experiments must first align effective action and observation windows and validate numerical output across processing paths.

The source inspection motivates concrete implementation improvements: persistent thermal constraints across normal/recovery paths, per-target model support, explicit effective-action credit, sensor-age handling, safe cancellation during worker replacement, and fail-closed evidence writes. These are proposed repairs to the inspected snapshot, not changes retrospectively applied to the campaign. New-build results should form a separately identified experiment rather than silently replacing unfavorable historical runs.

7 Threats to validity

Internal validity. The recordings are not randomized paired trials. Different binaries, starting temperatures, unrecorded ambient conditions, live input and scheduling state can affect results. Sharing a configuration hash and nominal WARNING threshold does not make whole policy paths identical. GPU/EMC state and actual thread placement are incompletely observed. The reference source is not cryptographically bound to either historical executable; its defects identify plausible risks but cannot automatically explain every recorded outcome.

Construct validity. CPU/GPU output equivalence is unverified, with concrete discrepancies in the reference source. Throughput, processing latency, whole-mission energy and sampled power have different boundaries and denominators. Measured GA compute excludes other costs. Software readback is narrower than physical state, and denial compliance is narrower than thermal containment. Forecast intervals are assigned, settings-change status is unknown, and timestamp-safe baselines cannot repair missing acquisition provenance or overlapping pipeline work. Boundary exclusions reduce one identified error but do not make forecasts causal action labels.

Statistical validity. Two runs per condition permit transparent descriptive contrasts but limited population inference. Frames and candidate populations are autocorrelated within a run. Within-run bootstrap intervals do not establish reproducibility between runs; some phases remain trend-dominated. Composite score weights, metric dropping and forecast baselines are post hoc. Outcome-selected exclusions would bias the comparison, so all supplied runs remain visible.

External validity. The study covers one Jetson Nano, one camera/presentation pipeline, four workload implementations and fixed phase durations. Results do not establish device independence, higher-resolution behavior, scaling to additional accelerators or general workload transfer. The application supervisor coexists with vendor power/thermal mechanisms[10]; this study does not certify those mechanisms or a hard real-time system.

8 Conclusion

Across six dynamic and twelve static-policy missions, the evaluated controller preserves the logged generation counter through all 66 dynamic workload changes and retains workload observations on recurrence. Schedule-matched static comparisons reveal heterogeneous outcomes: a favorable aggressive Median trade-off against GPU_STATIC and BALANCED coexists with a faster CPU_STATIC centre and substantial inactive Gaussian/Median throughput deficits. Forecasts are useful in a retrospective selected-trajectory comparison, while recurrence does not show a consistent error reduction. The evidence supports persistent runtime operation and bounded interpretations of observed trade-offs. Equivalent-output tests, source/build binding and controlled mechanism interventions are required before claiming general optimization superiority or causal learning benefit.

Data and code availability

Online Resource 1 contains the mission inventory, channel definitions and supplementary tables. Online Resource 2 contains derived run, phase and candidate summaries, retained forecast pairs, analysis scripts, input-file hashes and the distributable members of the inspected reference source. Its reproducibility checks rebuild the reported summary calculations from preserved intermediate records; they are not a full replay of all raw streams.

The authors have assembled an eighteen-mission raw-data corpus for a new version of Zenodo record <https://doi.org/10.5281/zenodo.23350634>. The new version is published as <https://doi.org/10.5281/zenodo.23676764>. It comprises per-mission archives, a mission index, thermal contracts, a data dictionary, analysis code and file checksums; the record’s analysis driver was executed in a fresh directory from the eighteen archives on 26 October 2026 and reproduced every frozen derived table (log and comparison report in the record). Dynamic action and forecast streams are distinguished from static-policy records. Data are designated CC BY 4.0; MIT applies to the authors’ code, with third-party files governed by their accompanying notices. The public source archive (SHA-256 2b821879...fa28) contains 90 unchanged members of the reviewed 91-file snapshot (SHA-256 3648a042...470c); one unused header carrying a proprietary notice is omitted. The per-file manifest preserves the correspondence to the reviewed snapshot. Neither source distribution is cryptographically bound to the historical executables. Section S11 of Online Resource 1 states the verification boundary.

Declarations

Funding This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, and by the Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM).

Competing interests The authors declare that they have no competing interests.

Ethics approval and consent Not applicable. The study involves no human participants, animals or personal data; the camera input is a live laboratory scene used only as a workload source.

Author contributions A. S. Rodriguez: conceptualization, methodology, software, investigation, data curation, formal analysis, visualization, writing – original draft. J. R. H. Carvalho: conceptualization, supervision, resources, writing – review and editing.

Use of AI-assisted tools AI assistants were used during manuscript drafting, source inspection, analysis-script preparation and editorial revision. The recorded experiments were not generated by these tools. The supplementary package identifies revised calculations and preserves the underlying tables and checks. This disclosure describes assistance and does not substitute for the authors’ review of the manuscript, code, data and references.

Related manuscript

A companion manuscript, “Request Is Not Readback: Auditing a Runtime Controller’s Self-Reported Telemetry on a Jetson Nano”, was submitted separately to IEEE Embedded Systems Letters (manuscript IEEE-ESL-Oct-26-0876). Its reported status is submitted, not accepted. The

present study adds the replicated dynamic/static policy evaluation, selected-candidate analysis and the qualified forecast comparison. The cover-letter materials disclose this relationship for editorial assessment.

References

- [1] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.
- [2] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011. doi: 10.1016/j.swevo.2011.05.001.
- [3] Torsten Hoeffler and Roberto Belli. Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Association for Computing Machinery, 2015. doi: 10.1145/2807591.2807644.
- [4] Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. Producing wrong data without doing anything obviously wrong! In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 265–276. Association for Computing Machinery, 2009. doi: 10.1145/1508244.1508275.
- [5] Raghavendra Pradyumna Pothukuchi, Joseph L. Greathouse, Karthik Rao, Christopher Erb, Leonardo Piga, Petros G. Voulgaris, and Josep Torrellas. Tangram: Integrated control of heterogeneous computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 384–398. Association for Computing Machinery, 2019. doi: 10.1145/3352460.3358285.
- [6] Tegg Taekyong Sung, Jeongsoo Ha, Jeewoo Kim, Alex Yahja, Chae-Bong Sohn, and Bo Ryu. DeepSoCS: A neural scheduler for heterogeneous system-on-chip (SoC) resource scheduling. *Electronics*, 9(6):936, 2020. doi: 10.3390/electronics9060936.
- [7] Tegg Taekyong Sung and Bo Ryu. SoCRATES: System-on-chip resource adaptive scheduling using deep reinforcement learning. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 496–501. IEEE, 2021. URL <https://arxiv.org/abs/2104.14354>.
- [8] Joseph Zuckerman, Davide Giri, Jihye Kwon, Paolo Mantovani, and Luca P. Carloni. Cohmeleon: Learning-based orchestration of accelerator coherence in heterogeneous SoCs. In *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture*. Association for Computing Machinery, 2021. doi: 10.1145/3466752.3480065.
- [9] Rafael J. Wysocki. *CPU Performance Scaling*. The Linux Kernel Documentation, n.d. URL <https://www.kernel.org/doc/html/latest/admin-guide/pm/cpufreq.html>. Accessed 25 October 2026.
- [10] NVIDIA Corporation. *Power Management for Jetson Nano and Jetson TX1 Devices*. NVIDIA, n.d. URL https://docs.nvidia.com/jetson/archives/14t-archived/14t-3275/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_nano.html. Tegra Linux Driver Package Development Guide, L4T 32.7.5. Accessed 25 October 2026.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [ESM1.pdf](#)
- [ESM2.zip](#)