When to Wait, When to Intervene: Separating Hazard-Related Delays from Computational Congestion

Anonymous authors
Paper under double-blind review

Abstract

A safety check may be late because its task is difficult or because its processor is busy. When several checks share a processor, their joint silence can therefore mislead a monitor deciding whether to wait, approve a route, or request intervention. We study this problem using a shared-load observation model and measured checker executions on simulated drone routes. Controlled replays preserve every checker's risk-conditional completion-time distribution while changing dependence between checks; this alone reverses the relative accuracy of shared-load and independent risk estimates. Received verification progress reduces integrated pending-risk Brier score by 18.1%. In a separate test of 24,576 executions, a rule based on CPU-sample arrival reduces hazardous approvals by 5.76 percentage points at the cost of more interventions. Combining clear segments returned by complementary checks permits earlier geometric approval with zero observed unsafe approvals: recorded stopping time falls by 74.1 ms on average and 1.50 ms at the median. The results show why marginal runtime models are insufficient for evaluating a safety monitor, and how partial verification evidence changes its safety, waiting, and intervention tradeoffs.

1 Introduction

Two safety checks are running, and neither has answered. Their silence could mean that the route is difficult to assess. It could also mean that both workers are waiting for the same busy processor. A monitor that treats the delays as independent warnings may count the same bottleneck twice; a monitor that discounts all delay as congestion may miss useful evidence of danger. The distinction matters when the monitor must choose between waiting, approving the route, and requesting intervention.

We study this problem with geometric collision checks on simulator-derived drone routes. The label records whether a fixed piecewise-linear path intersects an inflated obstacle. Checker execution and background CPU workloads are measured, while the monitor sees only received verdicts, workload probes and, in separate experiments, progress or CPU samples. This setting exposes the interaction between task risk and computation without requiring physical flight.

Our contribution is a controlled evaluation of what pending checks tell a monitor and what acting on that information costs. First, we compare shared-load and independent observation models using the same fitted marginal curves. Timing replays then change cross-checker dependence while preserving each checker's risk-conditional completion-time distribution. Second, we add received progress and CPU observations to test whether more informative histories improve prediction and decisions. Third, we compare those probabilistic decisions with approval based on the union of verified route segments. Policies are selected on calibration data and evaluated on fresh executions, with waiting and intervention accounted for separately.

The experiments identify three distinct effects. Changing dependence alone reverses the relative prediction ranking of the shared-load and independent models. Received progress lowers integrated pending-risk Brier score by 18.1%, but that improvement does not by itself ensure fewer hazardous approvals. Finally, complementary clear segments permit earlier geometric approval than waiting for a complete verdict. The CPU-observation experiment provides a further decision tradeoff: fewer

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

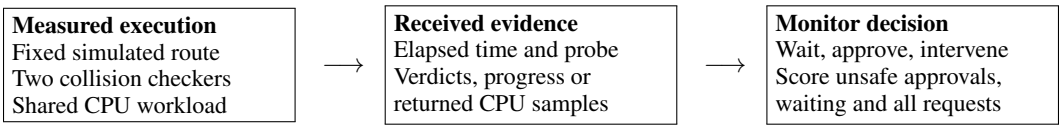

Figure 1: Experimental setup. The monitor uses only evidence received by its decision checkpoint. Hazard labels and assigned load are withheld from the policy and used for evaluation.

hazardous approvals accompany more interventions. Reporting these effects separately makes clear which benefit comes from inference, which comes from additional evidence, and which consumes more resources.

1.1 Why marginal checker behavior is insufficient

Consider a hazard label $H \in \{0, 1\}$ with prior $1/2$. Let $X_i = 1$ mean that check i remains pending at a fixed time. In each of two environments, retain $\mathbb{P}(X_i = 1 \,|\, H = 1) = 0.6$ and $\mathbb{P}(X_i = 1 \,|\, H = 0) = 0.4$ for both checks. In environment A, let $\mathbb{P}(X_1 = X_2 = 1 \,|\, H = 1) = 0.2$ and the corresponding safe probability be 0.4. In environment B, these joint probabilities are 0.6 and 0, respectively. Both joint tables preserve the stated marginals, yet

$$\mathbb{P}(H = 1 \,|\, X_1 = X_2 = 1) = \tfrac{1}{3} \quad \text{in A}, \qquad \mathbb{P}(H = 1 \,|\, X_1 = X_2 = 1) = 1 \quad \text{in B}. \tag{1}$$

An independence calculation returns $0.6^2/(0.6^2 + 0.4^2) = 9/13$ in either environment. Individual risk-conditional runtime distributions therefore cannot determine the risk associated with joint silence. This elementary construction motivates the experimental control; Appendix A gives the complete tables.

2 Methods

2.1 Decision problem and available information

We study a monitor that must approve a fixed candidate route, request intervention, or wait while two automated collision checks run. Let $H \in \{0, 1\}$ denote whether that route violates the benchmark's geometric predicate, L a shared computational-load regime, and C observed task complexity. The hidden state is static during a query in the initial model. A completed workload probe Q and the absence of checker verdicts convey evidence about L and H; later experiments additionally expose received progress or CPU observations. Intervention denotes a request with an assigned cost, not a measured human response or successful physical rescue.

All inference is restricted to information received by the monitor. Producer timestamps, eventual checker duration, evaluator labels, and assigned load are not available to the decision rule. A received complete verdict takes precedence: a clear verdict approves and a hazardous verdict requests intervention. These verdicts are checked against an independent geometric reference. This makes completion decisive for the specified static predicate, without implying general safety of the underlying vehicle.

For a policy π, approval indicator A_π, intervention indicator I_π, and total waiting time W_π, the fitted decision objective is

$$J(\pi) = \mathbb{E}[c_H H A_\pi + c_I I_\pi + c_T W_\pi]\,. \tag{2}$$

All intervention requests incur c_I, including necessary requests on hazardous routes. Preparation and probe latency are charged once in evaluation. They are sunk costs at the post-probe planning origin. Probability estimation, weighted decision cost, and performance under explicit resource constraints are evaluated separately.

2.2 A shared-load likelihood for pending checks

Let T_k be the received completion time of checker k, and write $S_{kh\ell}(t) = \Pr(T_k > t \mid H = h, L = \ell, C)$. The probe determines $\pi_\ell(Q, C) = \Pr(L = \ell \mid Q, C)$. Assuming conditional independence

of checker times given (H, L, C), the both-pending likelihood is

$$S_h^{\text{shared}}(t) = \sum_{\ell} \pi_\ell(Q, C) S_{1h\ell}(t) S_{2h\ell}(t). \tag{3}$$

Consequently, with registered hazard prior p_h,

$$b_t(h, \ell) = \frac{p_h \pi_\ell(Q, C) S_{1h\ell}(t) S_{2h\ell}(t)}{\sum_{h', \ell'} p_{h'} \pi_{\ell'}(Q, C) S_{1h'\ell'}(t) S_{2h'\ell'}(t)}. \tag{4}$$

Our central factorization control uses the same fitted marginal curves, probe likelihood, priors, and data, but replaces Eq. (3) by

$$S_h^{\text{ind}}(t) = \left[\sum_{\ell} \pi_\ell S_{1h\ell}(t)\right] \left[\sum_{\ell} \pi_\ell S_{2h\ell}(t)\right]. \tag{5}$$

The difference is the weighted covariance of the two survival probabilities across load states. We also include an information ablation without Q and a direct empirical joint-survival model conditioned on (H, C, Q). The latter permits residual checker dependence that the shared-load factorization does not explain. These are established probabilistic constructions; the empirical question is whether the distinction changes useful decisions.

Estimation uses training data only. The benchmark prior is $\Pr(H = 1) = 1/2$, matching its balanced construction, with a uniform prior over represented randomized load levels. Probe tertiles and nine positive elapsed-time knots come from training quantiles. Empirical survival curves use half-count tail smoothing and hierarchical shrinkage of 12 pseudocounts from complexity/load cells toward broader cells. Component curves interpolate linearly. Let $D(t) = \sum_h p_h S_h(t)$ and t' be the next knot. For two still-pending checks, backward induction compares

$$Q_{\text{approve}}(t) = c_H \frac{p_1 S_1(t)}{D(t)}, \qquad Q_{\text{intervene}}(t) = c_I,$$
$$Q_{\text{wait}}(t) = \frac{c_T \int_t^{t'} D(u)\, du + c_I p_1 \{S_1(t) - S_1(t')\} + D(t') V(t')}{D(t)}, \tag{6}$$

where $V(t) = \min_a Q_a(t)$. Here $S_1(t)$ denotes the both-pending survival conditional on hazard, not checker 1's marginal. Waiting ends at the next knot or the first received verdict. Piecewise-quadratic products are integrated exactly; waiting is unavailable at the finite horizon. This solves the fitted finite model, not an unknown real-world POMDP.

2.3 Received-history extensions and comparison policies

The progress and CPU-observation studies use empirical scenario trees over complete, coarsened received histories. Each hidden scenario specifies a hazard label, successive observations, and terminal completion. The belief is the normalized mass of training scenarios compatible with the observed prefix; the latest counter alone is not assumed sufficient. At a pending tree node n, let r_n be hazard probability, d_n expected time to the next event, u_n hazardous-completion probability, and p_{nm} pending-child probabilities. The continuation cost is

$$Q_{\text{wait}}(n) = c_T d_n + c_I u_n + \sum_m p_{nm} V(m). \tag{7}$$

The fitted transitions satisfy $r_n = u_n + \sum_m p_{nm} r_m$, so risk and continuation costs use the same joint distribution. At unsupported nodes, the progress study uses a specified timing fallback and includes its cost. The CPU-observation study instead forces intervention at sparse, unseen, or horizon-limited histories. Its optional label-neutral regularization modifies whole-scenario mass, including hazardous completions, before solving. It therefore changes the joint model rather than adjusting an approval threshold after fitting. Grouped training cross-validation selects the regularization strength and finite horizon. Progress certificates remain separate from synthetic label regularization.

The CPU-observation study compares wall/probe histories with histories augmented by returned per-checker CPU samples, and evaluates progress as a secondary interface. Its CPU candidate pool

also contains a context-specific threshold/deadline policy: a verdict takes precedence; otherwise, approve if both CPU samples have arrived and $\min(U_1, U_2)/C$ reaches the threshold, intervene at the deadline, or wait. A zero threshold makes this an availability rule rather than a test of CPU magnitude. Timing comparators include fixed and context-dependent stopping rules, the four survival models, and wall/probe scenario models. Immediate intervention and waiting for a verdict remain eligible references whenever feasible. The original survival plans execute a precommitted first stop at the next recorded opportunity; they do not continuously replan after checkpoint overshoot.

2.4 Measured benchmark, selection, and uncertainty

The paced studies generate drone trajectories with Crazyflow (Schuck et al., 2026) and retain their static, sampled route geometry. Two persistent FCL (Pan et al., 2012) workers check the same route in opposite traversal orders against 64 or 256 inflated spherical obstacles. A route group contains paired clear/hazardous variants of both complexities. CPU perception and voxel-processing workloads create contention; we record checker execution, message receipt, and workload measurements. Hazardous checks can terminate early, so longer delays need not imply greater danger. The progress and CPU-observation studies use an aggregate average pacing budget of 0.20 CPU cores and require no GPU. Policies are replayed at recorded controller opportunities, retaining preparation, receipt delay, and checkpoint overshoot. Online policy-inference cost is measured separately; it is not included in these recorded waits.

Related routes remain in the same split. The progress study uses 128 training, 128 calibration, and 256 held-out route groups. The CPU-observation study uses 128 training and 128 calibration groups, followed by 1,024 fresh test groups: 24,576 instrumented executions, including 12,288 hazardous ones, organized into 128 eight-group acquisition blocks. Initially resolved queries remain in the decision denominators. Source, data, model, selection, and analysis digests are frozen before the corresponding test; test outcomes do not select policies or resource levels.

For the CPU-observation study, the finite catalog contains 144 intervention/waiting-cost settings per modeled or contextual family, with deadline and threshold grids derived from training data. Calibration selects three common waiting caps and three common all-intervention caps from supported overlapping marginal resource ranges. Within each cell, feasible policies are ranked by hazardous-approval rate, then waiting, intervention rate, and stable identifier. A fixed ordering based only on feasibility selects the primary joint cell, starting with the middle cap on each resource. The CPU candidate is compared with the best eligible timing or reference policy on calibration data. We retain all infeasible cells and secondary frontiers. This comparison covers a finite catalog of pure policies; it does not optimize randomized mixtures or all possible controllers.

The primary criterion in the CPU-observation study requires a hazardous-approval-rate reduction of at least 0.002, a negative one-sided 95% upper bound on the paired difference, and both policies' one-sided waiting and all-intervention bounds below their common caps. The frozen caps are 120.0838 ms and 53.0206%. Common caps do not imply equal realized resource consumption. Paired acquisition-block Student-t intervals are supplemented by block-bootstrap and prespecified serial-dependence sensitivities. Unsupported flat empirical variances are not treated as zero-width guarantees. Prediction scores use their specified exposure weighting. The CPU-observation study also reports diagnostics at fixed pending-opportunity indices, because retrospectively reweighted histories do not represent ordinary online calibration. Budget analyses of the original study are post hoc and remain separate from this fresh test.

3 Results

The experiments examine how checker dependence changes risk inference, what received progress adds, and when these observations improve decisions under resource constraints. Each study uses its own evaluation protocol and comparison population.

Evaluation units and uncertainty. Routes come from simulated dynamics; safety-check runtimes and background workloads are measured executions. Unsafe approval denotes acceptance of a geometrically invalid route. Acquisition blocks capture dependence among repeated executions. The original study has two fresh confirmation cohorts, each with 256 route groups, 6,144 trials and 32

blocks. The progress study has 256 held-out groups and 12,288 trials, equally divided between reporting on and off, in 32 blocks. The CPU-observation study has 1,024 fresh test groups, 24,576 executions and 128 blocks. All benchmarks balance hazardous and safe cases. Policies and comparators are selected on separate calibration data before test acquisition.

3.1 Checker dependence changes the interpretation of pending results

In the original study, shared-load inference improves integrated pending-risk Brier score over independence with the same telemetry by 0.21% and 0.13% relative in confirmation A/B. Absolute differences are -0.00008394 and -0.00004642, with registered 97.5% block intervals $[-0.00011768, -0.00005353]$ and $[-0.00007554, -0.00001710]$. A directly estimated joint model scores better. The predictive gain does not establish lower decision cost than calibration-selected context stopping: differences are $+0.00819$ and $+0.00158$, with intervals $[-0.00211, +0.02070]$ and $[-0.00843, +0.01074]$. These intervals adjust two endpoints within each cohort, rather than all comparisons across cohorts. The separately frozen quiet-queue follow-up also fails its superiority criterion.

Controlled replays isolate the role of dependence. Within strata preserving label, complexity, session, partition, acquisition block, probe bin and assigned load, we preserve each checker's complete multiset of completion times and all verdicts while changing the association between checker times. With policies frozen, the latent-minus-independent Brier difference reverses sign between comonotonic and countermonotonic couplings in both cohorts (Figure 2). The specified marginal timing distributions therefore do not determine comparative prediction quality. This intervention changes recorded associations; it does not demonstrate a physical operation that changes dependence alone. Recoupling also changes the population and duration of both-pending observations, so the integrated score reflects both prediction error and pending exposure. Within each coupling, both models receive the same observations.

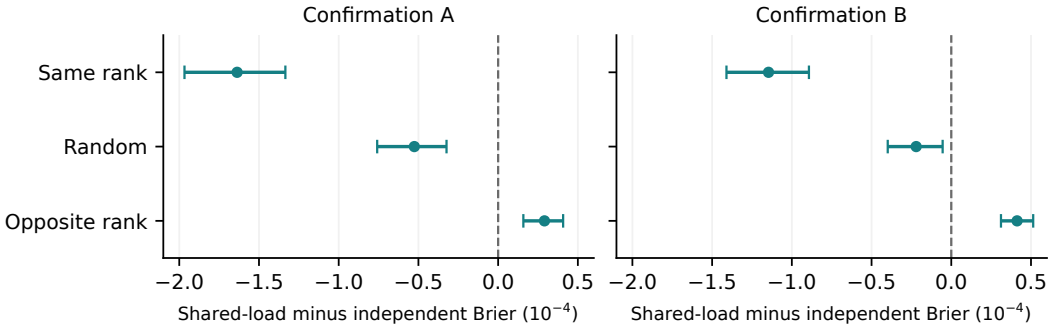

Figure 2: Changing dependence reverses the relative prediction ranking while preserving the specified per-checker timing marginals. Same-rank, random, and opposite-rank assignments are all shown for both original cohorts; negative differences favor shared-load inference. Bars are descriptive 95% block-bootstrap intervals with 2,000 resamples of 32 original blocks. These are constructed timing replays; pending exposure can differ across assignments.

An exploratory early-exit control shows that hazard-related delay can have either direction. Hazardous-to-safe median CPU-time ratios are 0.404/0.433 for 64/256 obstacles, versus 1.085/0.954 without early exit (24 development groups, 864 calls). A hazard can produce a *faster* verdict when its counterexample terminates the scan.

3.2 Received progress improves prediction and changes intervention use

On identical reporting-on observations, received progress reduces integrated pending Brier from 0.011514804 for the calibration-selected timing estimator to 0.009428477, an 18.1% relative reduction. The paired difference is -0.002086327, with descriptive 95% block interval $[-0.002809378, -0.001378493]$. This duration-weighted metric emphasizes long pending intervals in the registered grid; the reduction does not measure accident risk or performance at short deadlines.

Improved prediction does not establish the calibration and decision hypotheses. Progress ECE is 0.020876, 0.065635 and 0.066072 across loads; every upper limit exceeds the registered 0.02 tolerance. On the same reporting-on traces, progress approves 26 of 3,072 hazardous routes versus 11 for timing, while reducing unnecessary interventions from 1,066 to 378 among 3,072 safe routes. Its weighted cost is lower by 0.064499 (descriptive 95% interval $[-0.079803, -0.047874]$), reflecting a safety–intervention tradeoff. Against reporting-off timing, the registered comparison instead gives 26 versus 13 unsafe approvals and cost difference $+0.070378$ (adjusted 98.3333% interval $[-0.114817, +0.245388]$). Charging the different preparation costs changes the comparison: useful information alone does not establish a net instrumentation benefit.

3.3 Risk reduction and resource qualification in a fresh constrained test

The CPU-observation study selects a CPU-availability rule against a contextual stopping rule before one fresh test. All six selected CPU thresholds are zero: the candidate uses whether both CPU samples have returned, rather than their magnitudes. This tests an informative arrival event, rather than joint CPU-history inference or a POMDP solver.

Policies are replayed at actual received-data checkpoints. Charged waiting includes preparation, probe and collector waiting; online learned-policy inference and action publication are excluded. These costs characterize the recorded interface, rather than net live controller performance.

Measure	CPU availability	Context stopping	Common cap
Unsafe approvals / hazards	4,554/12,288	5,262/12,288	—
Unnecessary interventions / safe	5,105/12,288	0/12,288	—
All interventions / executions	12,839/24,576	7,026/24,576	—
Observed intervention rate	52.2420%	28.5889%	53.0206%
Intervention upper one-sided 95%	53.508%	29.366%	53.0206%
Mean paid wait (ms)	30.969	44.362	120.084
Wait upper one-sided 95% (ms)	33.140	47.160	120.084

Table 1: The preselected CPU-observation primary, evaluated on 1,024 route groups in 128 acquisition blocks. Waiting includes measured preparation/probe and collector waiting; offline policy inference is measured separately. Caps are common upper constraints, not equal realized consumption.

Unsafe-approval risk is 5.76 percentage points lower (95% paired acquisition-block Student-t interval $[-8.836, -2.688]$; one-sided upper limit -3.188). The joint success criterion also requires both policies' upper resource limits to satisfy both caps. The candidate's observed intervention rate is below the cap, but its upper limit exceeds it (Table 1); resource adherence is therefore not established and the preregistered joint criterion is not met. This is uncertainty about adherence, rather than an observed budget violation. The remaining hazardous approvals also rule out describing either policy as operationally safe.

All nine preselected joint-resource cells appear in Appendix B as unadjusted descriptive comparisons, with no test-based cell selection. "Strongest" refers to the evaluated finite deterministic-policy catalog; the study does not optimize stochastic mixtures or establish a global constrained optimum.

One preselected secondary operating point shows a resource-qualified benefit. With a 192.494 ms waiting cap and 47.912% intervention cap, the selected CPU-family rule approves 765 hazardous routes versus 891 for the contextual comparator, out of 12,288 hazards each. Both satisfy the recorded upper resource bounds; the paired risk difference is -1.025 percentage points (unadjusted descriptive 95% interval $[-1.383, -0.668]$). At the adjacent intervention cap, the comparison does not favor the CPU family. The benefit is confined to an operating region in this grid and does not replace the designated primary result.

3.4 Received clearance provides a distinct stopping signal

Measured CPU is not invariant to fixed verified work. In development, CPU per verified segment rises by factors of 1.513 and 1.547 at the two nonzero loads, with descriptive block intervals $[1.452, 1.574]$ and $[1.500, 1.591]$. CPU consumption is therefore not a pure hazard-work counter, and subtracting it from wall time does not causally identify congestion. Combined

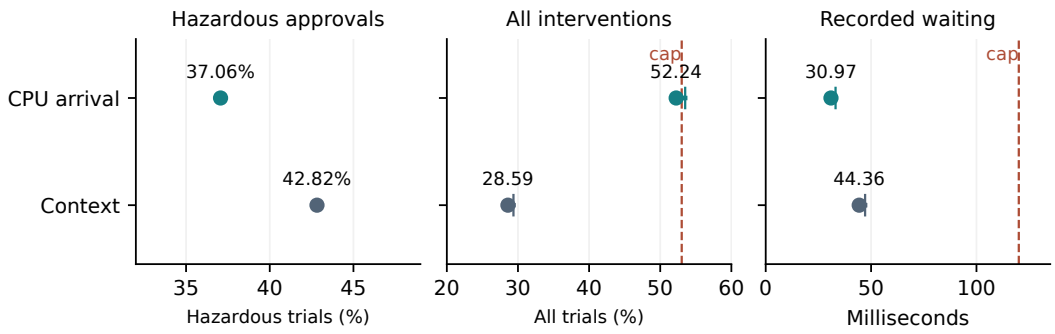

Figure 3: The designated primary operating point: fewer hazardous approvals and less recorded waiting accompany more intervention requests. Dots are observed means; the short rightward segments in the resource panels end at one-sided 95% upper bounds. Dashed lines show the common caps. The risk panel displays point estimates; the paired risk-difference interval is reported in the text.

polling/progress instrumentation has a mean first-verdict wall difference of 46.847 ms, with wide interval $[-70.045, 142.839]$ ms; these controls do not isolate progress-reporting overhead.

A fixed certificate-based reference approves only when received complementary clear frontiers cover every segment, or when a safe verdict arrives. On the same 24,576 traces it has zero observed unsafe approvals, as does waiting for a final verdict. Mean paid wait is 172.972 versus 247.072 ms. The descriptive paired-block mean reduction is 74.1 ms, with 95% interval $[70.165, 78.035]$ ms, but the median across all executions is only 1.50 ms and the 95th percentile is 13.091 ms. Long tails dominate the mean. This establishes conditional geometric evidence and offline stopping potential, rather than a new verification theorem, a population zero-risk guarantee or a measured live action-return saving. Both certificate/deadline families retain null calibration selections in all nine joint-resource cells.

4 Related work

Completion times and metareasoning. Decision-theoretic metareasoning values computation through its consequences for external actions (Russell & Wefald, 1991). Hansen & Zilberstein (2001) develop dynamic programming for monitoring and stopping anytime algorithms, accounting for uncertain progress and monitoring costs. The closest temporal-belief antecedent is Eddy et al. (2026), who formulate completion-time announcements as a mixed-observability Markov decision process, balancing announcement accuracy against update frequency. Coles et al. (2024) study concurrent planning and execution under wall-clock deadlines. Runtime-distribution learning also handles censoring: Tuero & Buro (2021) learn from completed runs and lower bounds on unfinished runs. Our empirical question concerns safety-relevant pending evidence under shared computation, with decision quality assessed under explicit resource constraints.

Dependent survival and probabilistic evaluation. Shared-frailty models formalize how a common unobserved factor induces dependence between event times (Oakes, 1989). Our observation model uses this source of dependence without assuming a proportional-hazards structure. Proper scoring rules assess probabilistic predictions (Gneiting & Raftery, 2007), while calibration and sharpness describe distinct forecast properties (Gneiting et al., 2007). These distinctions motivate separate checks of predictive score, reliability and downstream stopping outcomes: a lower aggregate Brier score alone does not establish calibrated hazard probabilities or better resource-constrained decisions.

Contention and partial progress. LATE prioritizes speculative execution using estimated remaining time in heterogeneous MapReduce environments (Zaharia et al., 2008); Mantri uses resource and progress information to mitigate different causes of stragglers (Ananthanarayanan et al., 2010). Execution-time budgets and blocking-time protection are established operating-system

mechanisms (AUTOSAR, 2024). Partial verification information is likewise established: Bayesian active edge evaluation selects expensive validity queries to identify feasible paths (Choudhury et al., 2018), while BUBAAK coordinates verification tools through shared artifacts (Chalupa & Richter, 2025). We hold the queried geometric content fixed and compare stopping rules using actually received evidence, connecting execution behavior to safety-relevant labels and approval outcomes.

Runtime assurance and evaluation. Runtime assurance separates a performance controller from a safety-preserving fallback. Black-Box Simplex uses runtime checks to support this separation without requiring a statically verified baseline controller (Mehmood et al., 2022). RTAEval provides a framework for evaluating assurance logic, including its computational behavior (Miller et al., 2023). Our recorded request for intervention is a decision outcome rather than a verified fallback maneuver. Similarly, the union of received clear segments certifies only the specified immutable geometric query under the checker assumptions; it is not a proof of closed-loop flight safety.

5 LIMITATIONS

The experiments concern decisions about cached, simulator-derived routes on one paced Mac. An unsafe approval violates a static geometric predicate; it is not an observed collision. An intervention records a request, without establishing that a person or fallback controller prevents harm. Balanced hazard prevalence, constructed obstacles, seen workload types and CPU pacing limit deployment interpretation. Pacing changes observation opportunities and latency tails, so measured milliseconds do not establish native drone deadlines or acoustic guarantees.

The shared-load model assumes fixed load within a query and conditional checker independence given a coarse state. Residual scheduling, task difficulty and resource competition can violate these assumptions; richer history models also face finite support and forced fallback. CPU time is not invariant geometric work, and wall time minus CPU time does not uniquely identify congestion. The selected CPU-availability rule uses zero thresholds, so its result concerns sample arrival rather than numerical CPU information or POMDP superiority. Complementary coverage provides a geometric certificate conditional on the checked predicate and correct receipts.

Common caps constrain policies without equalizing realized resource use. The finite deterministic catalog omits optimized randomized mixtures, and secondary comparisons remain descriptive. Approximate block intervals assume sufficiently stable independent blocks; serial sensitivity checks cannot establish that assumption. Favorable secondary results do not overturn the primary resource-qualification limitation, and zero observed certificate errors do not establish population zero risk.

Offline replay compares information under a shared recorded interface but excludes the full live inference-and-publication path. It therefore cannot establish net deployment speedup. Controlled timing replays, measured load experiments and exposed engineering extensions have distinct evidential roles; ongoing model-transfer and live-callback experiments are outside the completed results. The contribution is an implemented, auditable comparison of delay dependence and received evidence, with measured benefits and operating limits.

6 CONCLUSION

The meaning of a delayed safety check depends on how it shares computation with other checks. Holding individual runtime distributions fixed can reverse the ranking of competing risk models. Received progress improves prediction, while the arrival of CPU samples changes the tradeoff between hazardous approvals and intervention. Complementary clearance offers a separate route to earlier approval: use verified portions of the route as soon as they are received. These effects require different evidence and should be evaluated separately. The next deployment question is whether their benefits survive the cost of live inference and action delivery.

AI USE STATEMENT

Generative AI tools, including OpenAI Codex and Anthropic Claude, assisted with research ideation, literature discovery, hypothesis and protocol development, mathematical formulation, im-

plementation, debugging, experiment orchestration, interpretation, translation, and drafting and formatting this manuscript. AI tools also helped construct software that generates the simulated experimental data; these data are generated by the stated simulator and geometric procedures, rather than presented as physical flight observations. Verification includes executable checks, source and data hashes, separately written reconstruction of the primary decisions and metrics, and critical review of claims. These are within-project checks, not independent external replication. AI-generated explanations do not constitute experimental evidence. The authors remain responsible for the submission and its claims.

ETHICS STATEMENT

The study uses simulated trajectories and local computation; it does not involve human participants, physical flight, or deployment of an intervention policy. A request for intervention is a recorded decision, not a measurement of human reaction or successful rescue. Unsafe geometric approvals are deliberately retained as outcomes of offline policy evaluation. They are not executed on a physical drone. The released interpretation is conditional on the specified checker and geometry semantics and must not be used as a general flight-safety guarantee.

REPRODUCIBILITY STATEMENT

The experimental phases retain protocols, source and input hashes, simulator geometry, raw received-event traces, calibration choices, all candidate outcomes, and independently reconstructed primary decisions. The appendix specifies the clock boundaries, resource criteria, and negative results needed to interpret the reported comparison. A supplementary archive contains anonymized code and aggregate evidence with a manifest. This archive is a review artifact; no public repository or external replication is claimed.

REFERENCES

Ganesh Ananthanarayanan, Srikanth Kandula, Albert Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris. Reining in the outliers in Map-Reduce clusters using Mantri. In *9th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 2010. URL https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Ananthanarayanan.pdf.

AUTOSAR. *Specification of Operating System*. AUTOSAR, 2024. URL https://www.autosar.org/fileadmin/standards/R24-11/CP/AUTOSAR_CP_SWS_OS.pdf. Classic Platform R24-11, Document ID 34.

Marek Chalupa and Cedric Richter. BUBAAK: Dynamic cooperative verification. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 15698 of *Lecture Notes in Computer Science*, pp. 212–216. Springer, 2025. doi: 10.1007/978-3-031-90660-2_14. URL https://link.springer.com/chapter/10.1007/978-3-031-90660-2_14.

Sanjiban Choudhury, Siddhartha Srinivasa, and Sebastian Scherer. Bayesian active edge evaluation on expensive graphs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 4890–4897, 2018. doi: 10.24963/ijcai.2018/679. URL https://www.ijcai.org/proceedings/2018/679.

Andrew Coles, Erez Karpas, Andrey Lavrinenko, Wheeler Ruml, Solomon Eyal Shimony, and Shahaf Shperberg. Planning and acting while the clock ticks. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pp. 95–103, 2024. doi: 10.1609/icaps.v34i1.31465. URL https://ojs.aaai.org/index.php/ICAPS/article/view/31465.

Duncan Eddy, Esen Yel, Emma Passmore, Niles Egan, Grayson Armour, Dylan M. Asmar, and Mykel J. Kochenderfer. Optimizing task completion time updates using POMDPs, 2026. URL https://arxiv.org/abs/2603.12340. arXiv preprint.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(2): 243–268, 2007. doi: 10.1111/j.1467-9868.2007.00587.x. URL https://academic.oup.com/jrsssb/article/69/2/243/7109375.

Eric A. Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1–2):139–157, 2001. doi: 10.1016/S0004-3702(00)00068-0. URL https://www.sciencedirect.com/science/article/pii/S0004370200000680.

Usama Mehmood, Sanaz Sheikhi, Stanley Bak, Scott A. Smolka, and Scott D. Stoller. The Black-Box Simplex architecture for runtime assurance of autonomous CPS. In *NASA Formal Methods*, pp. 231–250. Springer, 2022. doi: 10.1007/978-3-031-06773-0_12. URL https://arxiv.org/abs/2102.12981.

Kristina Miller, Christopher K. Zeitler, William Shen, Mahesh Viswanathan, and Sayan Mitra. RTAEval: A framework for evaluating runtime assurance logic. In *Automated Technology for Verification and Analysis*, volume 14216 of *Lecture Notes in Computer Science*, pp. 302–313. Springer, 2023. doi: 10.1007/978-3-031-45332-8_17. URL https://mitras.ece.illinois.edu/research/2023/RTAEval_ATVA23.pdf.

David Oakes. Bivariate survival models induced by frailties. *Journal of the American Statistical Association*, 84(406):487–493, 1989. doi: 10.1080/01621459.1989.10478795. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478795.

Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012. doi: 10.1109/ICRA.2012.6225337. URL https://ieeexplore.ieee.org/document/6225337.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1–3):361–395, 1991. doi: 10.1016/0004-3702(91)90015-C. URL https://www.sciencedirect.com/science/article/pii/000437029190015C.

Martin Schuck, Marcel P. Rath, Yufei Hua, Abhishek Goudar, SiQi Zhou, and Angela P. Schoellig. Crazyflow: An accurate, GPU-accelerated, differentiable drone simulator in JAX, 2026. URL https://arxiv.org/abs/2606.01478. arXiv preprint.

Jake Tuero and Michael Buro. Bayes DistNet: A robust neural network for algorithm runtime distribution predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12418–12426, 2021. doi: 10.1609/aaai.v35i14.17473. URL https://ojs.aaai.org/index.php/AAAI/article/view/17473.

Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving MapReduce performance in heterogeneous environments. In *8th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, 2008. URL https://www.usenix.org/legacy/event/osdi08/tech/full_papers/zaharia/zaharia_html/.

A Complete dependence construction

Table 2 supplies the joint distributions used in the introduction. Each row sums to one. In both environments, each checker has pending probability 0.6 on hazardous tasks and 0.4 on safe tasks. Only the dependence differs. The construction concerns a fixed time and binary pending observations; it does not assert a measured hardware effect, a new theorem, or a complete dynamic load model.

Table 2: Illustrative conditional joint probabilities for pending indicators.

Environment	Hazard label	$(1,1)$	$(1,0)$	$(0,1)$	$(0,0)$
A	$H=1$	.2	.4	.4	0
A	$H=0$	.4	0	0	.6
B	$H=1$	.6	0	0	.4
B	$H=0$	0	.4	.4	.2

B ALL PRESELECTED JOINT-RESOURCE COMPARISONS

The nine comparisons in Table 3 are descriptive secondary results from the same complete frozen test. Policies in each cell are selected on calibration, not by choosing the favorable TEST row. The designated primary is the middle wait/intervention cell. Confidence intervals are unadjusted for multiple comparisons, and secondary comparisons do not replace the designated primary analysis. A budget qualification requires the recorded upper one-sided 95% limits to meet both resource caps; an unqualified result is not necessarily a point-estimate budget violation.

Table 3: Complete preselected joint-resource cells. Risk difference is CPU family minus the CAL-selected timing/reference comparator, in percentage points. C/B denotes candidate/comparator qualification under both caps.

Wait cap(ms)	Request cap(%)	Risk difference	95% interval	C/B qualified
67.467	47.912	-8.830	$[-11.782, -5.878]$	No/Yes
67.467	53.021	-8.830	$[-11.782, -5.878]$	No/Yes
67.467	69.227	0.277	$[0.043, 0.510]$	No/No
120.084	47.912	-5.762	$[-8.836, -2.688]$	No/Yes
120.084	53.021	-5.762	$[-8.836, -2.688]$	No/Yes
120.084	69.227	0.041	$[-0.001, 0.083]$	No/No
192.494	47.912	-1.025	$[-1.383, -0.668]$	Yes/Yes
192.494	53.021	0.480	$[-0.075, 1.035]$	Yes/Yes
192.494	69.227	0.041	$[-0.001, 0.083]$	No/No

The one favorable qualified secondary cell has a 192.494 ms waiting cap and 47.912% request cap. It compares a different CPU-family rule with a different timing rule from the primary. It is therefore a scoped descriptive operating point, not proof of a universal CPU signal, load invariance, or joint-model superiority. At the adjacent intervention cap with the same waiting cap, the selected CPU-family rule does not improve the point estimate relative to its selected comparator.

C RECORDED CLEARANCE VERSUS LIVE EXECUTION COST

The fixed received-frontier reference approves only after complementary checked portions cover the specified path, or after a safe verdict is received. The obstacle representation already includes the drone envelope. The numerical geometry oracle and checker semantics must agree; adding a second envelope would change the problem. A hazardous verdict always takes precedence over permissive pending prediction.

On the full frozen test, the clearance and final-verdict references each record zero unsafe approvals among 12,288 hazardous executions and 12,288 intervention requests among 24,576 executions. Their recorded mean waits are 172.972 ms and 247.072 ms. The paired mean difference is 74.100 ms, with descriptive interval $[70.165, 78.035]$ ms. The median is 1.502 ms, the 95th percentile 13.091 ms, and the maximum 2864.489 ms over all executions. These quantities are consistent with a small fraction of very large savings dominating the mean. They must not be presented as typical per-flight speedups or as actual cancelled-checker compute savings.

Neither family has a selected feasible policy in any of the nine joint-resource calibration cells. The fixed full-clear reference, one-resource comparisons, and full deadline curves remain separate sec-

ondary evidence. There is no replacement of an infeasible calibration selection with a favorable point chosen from TEST.

D INFORMATION, TIMING, AND REPRODUCIBILITY BOUNDARIES

A raw measurement has several relevant times: the worker's emission time, receipt by the monitor, completion of validation, the policy checkpoint, and return of the action callback. A value can become a policy input only after it has been received and is available at that checkpoint. Final CPU totals do not retroactively become intermediate measurements. Assigned worker load, geometric collision labels, and collision positions remain outside the policy observation.

The completed CPU study prices collector preparation/probe and recorded waiting. Its separate cached-history inference benchmark is not a live incremental monitor benchmark and is not included in the collector-clock resource bound. Later engineering explicitly instruments outer entry through callback publication, but those new ongoing experiments do not contribute confirmatory efficacy results to this manuscript. Likewise, the original timing-coupling replay preserves declared checker marginals, while real CPU-load interventions can change those marginals. The two operations test different mechanisms.

Source and input digests, complete candidate arrays, every preselected resource cell, failed clock checks, and corrected versions are retained. For source-code cross-reference, the primary CPU-availability rule is `cpu_threshold:127` and contextual stopping is `context_stop:113`. The primary reproduction rederives 49,152 policy decisions and their recorded waits on 24,576 executions; a separate saved-array check covers 1,658 candidate columns. This is independent code within the same research project, with shared data and assumptions. Secondary clearance computations reuse their declared implementations and do not inherit the stronger independent-primary reproduction claim.

The supplement provides a minimal anonymous path for checking the reported aggregate quantities and the dependence construction. It does not contain physical-flight evidence or establish external replication. Full execution logs and environment-specific provenance remain in the local research record; distribution of all raw traces is distinct from providing these aggregate review artifacts.