Harbor Task-Corpus Evaluation — Batch Report

Conductor-orchestrated agent harness vs. a 15-task Harbor (Terminal-Bench evolution) corpus
Sessions S23–S24, 2026-07-20 · Standard-shape ladder complete (10 of 15 tasks); 5 milestone tasks pending

1. Executive summary

We ran ten standard-shape Harbor tasks end-to-end through a multi-agent SDLC harness (frozen acceptance contracts → sandboxed builder → blind executing critic → mechanical verifier → grade against a pre-calibrated ground truth). Headline numbers:

- **Benchmark's own scoring (final artifact, `reward.txt`): 9 / 10 tasks at reward 1.**
- **Harness-certified (benchmark green AND independent critique passed): 8 / 10.** One task (`boa-eval-delete-v5`) scored reward 1 on the hidden suite but was *refused* by our critique layer with executed counter-evidence; one task (`abi-break-variant-ladder`) failed honestly at reward 0 under a deliberate low-cost-model exploration policy.
- **Three of the nine green results required exactly one critique-directed rework each** inside the preregistered two-attempt budget; every bounce converged in a single round.

The most important output for a benchmark author is not the score line — it is the **suite-quality evidence**. Within one batch, independent executing critique demonstrated that hidden suites mis-grade in *both* directions:

- **Benchmark-green-but-wrong, twice** (`boa-eval-delete-v5`, `archive-extract-hardlink-symlink`): artifacts passed the full hidden suite while violating documented task semantics, provable with small, format-legal counter-examples the fixtures never distinguish.
- **Benchmark-red-but-right, once** (`buddy-allocator-simulator`): a functionally correct artifact scored reward 0 because a textual self-containment scan matched a source *comment*.

Details, reproductions, and per-task write-ups follow. Section 5 is written specifically against Harbor's own validation checklist ("all behavior in instruction.md has corresponding tests").

2. Corpus and method

2.1 Corpus

15 Harbor tasks (format version 2.0), each shipped as `instruction.md / task.toml / environment/ (digest-pinned Dockerfile) / solution/solve.sh / tests/(test.sh + test_outputs.py)`. 10 are single-shot ("standard shape"), 5 are milestone-shaped (`steps/milestone_N/`). All 16 `task.toml` files in the corpus declare `allow_internet = false` — verified corpus-wide before the batch. Every task run

here is difficulty `hard` per its metadata. Verdicts were read exclusively per Harbor semantics: `/logs/verifier/reward.txt` (with `reward.json` fallback), never `test.sh` exit codes.

2.2 Grader calibration — discrimination proof before use

No suite was used for grading until it passed a fail-closed calibration: run the benchmark's own hidden suite in its pinned container against (a) the pristine shipped stub — must yield reward 0 — and (b) the benchmark's own oracle (`solve.sh`, mounted at the canonical `/oracle`) — must yield reward 1. All ten standard tasks calibrated DISCRIMINATING. Two oracle defects surfaced during calibration (Section 5.4).

2.3 Execution lifecycle per task

1. **Frozen contract.** Objective, inputs, and three mechanical evaluators (deliverable presence; containerized hidden-suite run; reward check) are bound and content-hashed (preregistration SHA-256) before any builder sees the task. The instruction.md text rides in verbatim as domain context.
2. **Builder.** A sandboxed headless CLI agent (primarily grok-4.5) works in an isolated scratch git repository seeded with only builder-visible materials. It can build/run inside the task's pinned Docker image but *cannot execute the hidden suite* (see 2.4).
3. **Sweep.** The conductor audits the builder's full session transcript for contamination (two modalities — see 2.4), commits the work, and registers the candidate artifact (content-addressed).
4. **Blind critique (the load-bearing layer).** A model-diverse critic (Claude Sonnet) that never sees the builder's account judges the artifact against the frozen contract. It must *execute*: re-run all three evaluators, exercise the real workflow, and — mandatory since the boa finding — design and run its **own variants** with outcomes predicted *before* execution (hand-computed values, independent mini-reference implementations, external reference engines). The hidden suite is treated as one finite-coverage evaluator, never as a critique substitute.
5. **Bounce protocol.** On a critique FAIL within the preregistered budget (`max_attempts=2`), the builder receives a rework brief naming the violated *documented* rule, the offending code sites, and a hand-verifiable acceptance case — never suite internals. The same critic then re-verdicts, re-running its own reproductions.
6. **Verify & grade.** A conductor-run mechanical verifier re-executes the evaluators fresh; the final grade is the suite reward vs. the calibrated ground truth of 1. Every state transition, cost, verdict, and grade is journaled in a hash-chained event log (chain verified, ~18k events).

2.4 Clean-room integrity design

Because these are real benchmark tasks (several derived from real projects with published fixes), contamination control was engineered explicitly, and it caught two real incidents (Section 6):

- **Workspace isolation:** builder workspaces live in a sibling-free directory tree; no path in or beside the workspace leads to `solution/`, `tests/`, or calibration records.
- **Material locks:** `solution/`, `tests/`, calibration dirs are `chmod 000` from builder launch; tests are unlocked only for critique/verify and everything is restored after grading.

- **Opaque suite resolution:** evaluator commands in contracts/briefs reference `$BEVAL_SUITE_DIR` / `$BEVAL_LOGS_DIR`, resolved only inside the verifier's process — no suite path ever appears in anything a builder can read.
- **Offline rule:** every brief forbids web retrieval outright. This mirrors the corpus's own `allow_internet = false` contract — with the important difference that the benchmark's network boundary encloses the task container, while an agent harness must also enclose the *agent*, which runs outside that container.
- **Transcript audits:** before any sweep, the builder's full session transcript is scanned for both leak modalities: benchmark-material access (paths, oracle mentions) and network retrieval (search, fetch, upstream PRs). A contaminated attempt is invalidated: no grade, no trust delta, spend still journaled.

Honest limits, stated plainly: builder and verification materials share one OS user on one machine, so the clean-room is enforced by permissions, path opacity, sandbox policy, and post-hoc transcript audit — strong layered evidence, not a cryptographic proof. A separate-uid/container design was scoped for the remaining tasks but not yet deployed.

3. Results

Task	Type	Benchmark reward (final artifact)	Harness verdict	Attempts	Hidden suite (final)	Notes
bathroom-capability-gate	build-from-spec (C++)	**1**	**CERTIFIED**	1	17/17	First task under the decontaminated design; critic's own input mutation hand-verified.
1c87d9a3 (billing module)	build-from-spec (module)	**1**	**CERTIFIED**	1	20/20	Critic constructed an input discriminating a local-date-vs-UTC rule with hand-computed numbers.
boa-eval-delete-v5	fix (real Rust codebase)	**1**	**REFUSED — green-but-wrong**	2*	4/4	Suite green; critique refuted generality vs. an external reference engine. *Attempt 1 invalidated for web contamination (Section 6).
ad-attribution-reconciler	build-from-spec (Go)	**1**	**CERTIFIED**	1	30/30	Two hand-computed critic variants (revision/void/seq ties; weighted remainder) matched exactly.
buddy-allocator-simulator	build-from-spec (Rust)	**1**	**CERTIFIED after 1 bounce**	2	34/34	**Red-but-right:** attempt 1 reward 0 (33/34) — a *comment* tripped the textual self-containment scan.
bplus-tree-replay	build-from-spec (Go, in-image)	**1**	**CERTIFIED**	1	35/35	Critic wrote an independent reference implementation; hand-traced variant matched incl. hash-chain digests.
apk-repository-index-audit	fix (synthetic Go tree)	**1**	**CERTIFIED**	1	15/15	Builder correctly ignored planted decoy config files (Section 5.3); determinism byte-verified.
brew-kettle-boil-ledger	fix (Go CLI)	**1**	**CERTIFIED after 1 bounce**	2	21/21	Attempt 1: double-rounding vs. documented raw-basis formula; critic hand-computed the discriminating case both directions.

archive-extract-hardlink-symlink	fix (C++ tool)	**1**	**CERTIFIED after 1 bounce**	2	11/11	**Green-but-wrong:** attempt 1 passed the full suite while violating two documented contract rules (Section 5.2).
abi-break-variant-ladder	fix (C++ cluster)	**0 (13/23 red)**	**FAILED (honest)**	2	10/23	Deliberate exploration route to a no-execution builder lane (Section 7.3); critic located the root cause in an untouched stub.

"CERTIFIED" = benchmark reward 1 *and* blind executing critique PASS *and* fresh mechanical verification green. Every grade was taken against a pre-proven discriminating grader (stub 0 / oracle 1). Two additional S23 task records exist as invalidated/failed pre-execution entries (a material-leak invalidation and a routing mis-fire caught before launch) — both journaled, neither graded; they are process records, not benchmark results.

3.1 How to read the score

If you evaluate us purely as a Harbor leaderboard entry: **9/10 on the standard ladder at difficulty hard**, with the tenth an intentional exploration cost (a cheap, execution-less model lane on the hardest task — its trust ledger now carries that datapoint with the confound noted). If you evaluate the harness as a quality instrument, the more meaningful claim is: **zero uncertified artifacts shipped** — every reward-1 artifact either survived independent adversarial execution or was refused despite its green suite.

4. The bounce protocol in practice

All three bounces converged in exactly one rework round:

Task	Attempt-1 defect (critique-found)	Rework size	Re-verdict evidence
buddy-allocator-simulator	Suite's self-containment test regex-scans the whole source; the comment `// No std::process::exit; just return.` matched `std::process`. Functionally correct: critic's five hand-computed variants all exact.	2 lines (comment reword + dropped unused `mut`)	Commit-to-commit diff proven non-functional; 34/34; code-path-adjacent variant re-run byte-exact.
brew-kettle-boil-ledger	Double rounding: intermediates stored round3'd while `numeric_rules.md` computes rates from raw values, rounding only the result (+0.001–0.002 drift; 4/21 red). Critic's own hand case: expected 5.786, emitted 5.787.	3 files, ~15 lines (raw basis through rate math; round3 at display/final only)	21/21; hand case exact at 5.786; atomicity + stale-revision spot-checks unregressed; reruns byte-identical.
archive-extract-hardlink-symlink	Two documented-contract violations invisible to the suite (Section 5.2).	2 files (verify comparand fix; new-only-path rollback on rejection)	Critic re-ran its own reproduction + control archives; fresh invalid-upgrade probe; zero orphans; suite still 11/11.

The rework briefs deliberately contained: the violated *documented* rule, the offending code sites, and one hand-verifiable acceptance case — and deliberately excluded suite internals (test names, pinned values). Builders converged without them every time. We consider this the harness's core competitive claim: the critique layer converts benchmark noise (both directions) into signal at a marginal cost of one rework round.

5. Suite-quality findings (the section written for a Harbor author)

Harbor's validation checklist requires that "all behavior in instruction.md has corresponding tests." Three results in this batch bear directly on that item. All three were found by the same mechanism: an executing critic obligated to design its own variants with predictions committed before running.

5.1 Green-but-wrong #1 — boa-eval-delete-v5 (S23)

- Builder's fix passed the full hidden suite (4/4, reward 1) on a calibrated grader.
- Critic's self-designed variant: property removal + type inspection inside one dynamic-code evaluation, then redeclare-and-read in a later evaluation. The patched engine raised a spurious `ReferenceError`; the V8 reference behavior is `true, undefined, 42`. Isolated across 8 variant runs; 3 pre-existing test failures were attributed to the base revision via byte-exact revert, so the regression is unambiguously the patch's.
- Disposition: we recorded the benchmark reward (1) and the harness refusal separately, and did not certify. The instruction documents the removal semantics the variant exercises; the shipped suite's cases simply never combine them.

5.2 Green-but-wrong #2 — archive-extract-hardlink-symlink (S24)

- Attempt 1 passed the full hidden suite (11/11, reward 1).
- **Defect A (wrong comparand):** the extraction contract says a hardlink "must share the target member's mode and modification time." The artifact's `verify()` instead validated the hardlink's on-disk state against the *hardlink record's own* declared mode/mtime. A format-legal archive whose `H` record declares 0600 against an `F` target declaring 0640 is spuriously rejected (exit 2, `final-state-mismatch`) even though the extracted filesystem state is exactly correct (inode shared, target's mode). A control archive identical except for matching declared fields succeeds — a minimal isolation pair. **Every shipped and suite fixture happens to declare identical H/F mode+mtime**, which is precisely why the suite stayed green.
- **Defect B (failure-path orphan):** that spurious rejection fired after materialization with the journal never advanced; a subsequent re-run of the prior generation then stranded an object owned by no completed generation — the exact "stale objects" class the instruction.md names as the defect to fix.
- Disposition: bounced; the rework (target-field comparison + symlink-safe rollback of new-only paths on pre-durable-journal rejection) survived the critic's own reproduction, control, and a fresh genuinely-invalid-upgrade probe. Certified.
- *Suggested suite addition:* one two-generation fixture pair whose H and F records declare differing mode/mtime, plus one rejected-upgrade-then-rerun scenario asserting no undeclared objects survive.

5.3 Red-but-right — buddy-allocator-simulator (S24), plus a trap observation

- Attempt 1 scored reward 0 with 33/34: the failing test scans the entire `main.rs` text (case-insensitive) for forbidden-API patterns, and matched the literal string `std::process` inside a comment explaining that the code deliberately does *not* call it. All five of the critic's hand-computed behavioral variants (multi-level coalescing, shrink/grow-in-place, relocate-OOM state preservation, 64-bit boundary at M=60, relocate free-list sourcing) matched exactly, and a 600k-operation trace ran in 0.161 s — the artifact was right; the grader's textual enforcement was over-broad. *Suggested fix:* strip comments/strings before the forbidden-pattern scan, or scan the AST.
- Related observation from `apk-repository-index-audit`: the tree ships config files (`priority_policy.toml`, `arch_defaults.json`, `sig_labels.json`) that **no runtime code reads**, whose contents contradict the normative contract document (e.g. a label mapping that would break the documented status enum). Our builder ignored them and implemented the document; the critic independently confirmed wiring them in would have failed the contract. If these are intentional traps, they work; if not, they are misleading debris.

5.4 Oracle and packaging defects found during calibration

- **brew-kettle-boil-ledger's `solve.sh` cannot run as shipped** under the canonical `/oracle` (or legacy `/solution`) mount: after `cd "$(dirname "$0")"` it runs `go build /app/environment/cmd/kettlectl` from outside any Go module, which fails with "go.mod file not found." The task's own suite rebuilds with `cwd=/app/environment` — we calibrated with a cwd-corrected equivalent (same patches, same build target) and it produced reward 1.
- **Mount-path convention split:** several oracles hardcode `/solution` (ad-attribution, buddy) while the spec's canonical mount is `/oracle`; others are path-relative. We mount both; worth normalizing in the corpus.
- Pre-existing base-revision test failures in the boa checkout (3) — worth a note in that task's metadata since they complicate naive suite-delta grading.

6. Integrity incidents — what the clean-room actually caught

Two real contamination events were caught and invalidated (no grade, no trust credit, spend journaled):

1. **Disk leak (S23, first task attempt).** An early brief rendered evaluator Docker mounts verbatim (exposing the suite path) and the workspace sat inside the benchmark record directory with `solution/` readable — the builder's sandbox blocked writes, not reads, and its transcript showed it reading the reference solution. The attempt was invalidated and the entire decontamination design of Section 2.4 was built in response, then re-validated on a fresh task instance.
2. **Network leak (S23, boa attempt 1).** The builder web-searched the symptoms, found the real upstream pull request — the ground-truth patch file in `solution/` is literally named after it — and applied the fetched diff. Disk locks held; the leak went *around* them via host-side web tooling. This is the instructive one for Harbor: `allow_internet = false` governs the task container, but an agent harness runs the agent outside that boundary — the offline contract must be enforced at the agent layer too. The offline rule + dual-modality transcript audit came out of this; boa attempt 2 ran fully clean.

Additionally, the material locks were exercised from the defender side: one critic attempted a minimal `chmod` on a locked directory when a conductor process error (see 7.4) left tests locked at critique time; the environment's permission layer denied it, the critic reported the block honestly rather than working around it, and the conductor corrected the state.

7. Agent-performance analysis

7.1 Builder lane (grok-4.5, headless CLI, sandboxed)

- 9 of 10 standard tasks were built by grok-4.5, routed on accumulated trust. Final-artifact quality after harness process: all 9 reached benchmark reward 1.
- First-attempt quality: 5 of 9 were certified with no rework; 3 needed one directed rework; boa's clean attempt was refused on generality despite suite green.
- Transport behavior worth knowing: the CLI frequently terminates with a "Cancelled" stop reason *after* its writes have landed (observed on roughly half the runs, including two completed reworks). We treat "Cancelled + writes landed" as a completion class and grade by effects; "Cancelled + zero writes at turn ~3" is a distinct stateless-early death (2× observed, both triggered by bare docker read commands under its sandbox — worked around by provisioning in-image docs into the workspace read-only).
- Measured tool-reported builder cost for the seven S24 standard tasks (nine runs incl. two reworks): **≈ $2.65 total**. Builder token totals ranged 73k–796k per run.

7.2 Critique lane (Claude Sonnet, blind, executing)

- Seven critique engagements (10 verdict rounds incl. 3 re-verdicts): every verdict was backed by fresh execution; 61k–277k tokens per engagement (Agent-tool measured).
- Verdict-deciding evidence was *never* the hidden suite alone: hand-computed expectation tables (ad-attribution, brew), an independently written reference implementation plus third-party digest check (bplus), format-legal counter-example archives with a minimal control pair (archive), an external reference engine (boa, S23), pre-registered predictions on container-copied packs (apk).
- Three critiques found acceptance-blockers that the calibrated hidden suites scored green or that self-checks missed; one critique correctly overrode a red suite. This is the empirical case that critique arms must *execute and design variants*, not read diffs.

7.3 Exploration datapoint (gemini-3.1-pro via a no-execution CLI lane)

- Our routing policy reserves a fraction of tasks for labeled exploration of off-ladder executors. It fired on the hardest task (`abi-break-variant-ladder`) and routed to a lane that can read and write files but **execute nothing** (no compiler, no shell).
- Outcome, honestly recorded: attempt 1 authored a 9-file repair blind; a conductor smoke build found one use-before-declaration compile error; the bounced attempt 2 fixed it (+3 lines) and self-audited its other files for the same class. The compiling artifact implemented determinism, sequencing, and facet weight-sensitivity genuinely (critic-verified) but missed a scope-lookup defect in an *untouched* stub — the kind of cross-module data-flow bug that plausibly requires execution feedback to find. Final: reward 0 (13/23 red, all on the same root cause the critic had isolated on three seeds).
- Cost of the datapoint: two attempts at unmeasurable token cost (the lane's CLI emits no usage; we record null, never fabricate), one critic engagement, and one task slot. The trust ledger now carries the first labeled datapoint for this executor with the difficulty confound noted.

7.4 Conductor process defects (ours, owned in-journal)

- A manual sweep for the non-standard builder lane skipped the tests-unlock step the scripted sweep performs — the critic was locked out of the suite evaluators, reported it honestly, and the conductor re-ran the suite post-hoc (result confirmed the critic's independent conclusion). Manual sweeps must replicate the scripted sweep's full postcondition list.
- One builder launch was wrapped in a nested background subshell, orphaning the process from task tracking (recovered via process-table inspection; cwd and sandbox were verified intact).
- Three stale-working-directory slips in conductor shell commands (caught immediately by failing path assertions; no state damage).

8. What remains

- **5 milestone-shaped tasks** (`steps/milestone_N/`), to run per the resolved design: one stateful workspace per task, N sequential harness lifecycles, per-milestone calibration (oracles 1..N–1 applied → milestone-N suite must read 0; adding oracle N → must read 1) and per-milestone grades.
- **Clean-room hardening** (scoped, not yet deployed): separate uid or container for builders, non-discoverable solution storage, and an OS-level file-access witness (`eslogger` / `fs_usage`) to upgrade "strongly evidenced" to "provable".
- **A consolidated suite-feedback note to the task authors** — Sections 5.1–5.4 are written to be lifted directly into that.

9. Appendix — traceability

Every claim in this report resolves to journaled events in a hash-chained log (verified intact, head seq 17951 at session close). Task instances: bathroom-capability-gate `task–a216ab3c967e` · billing module `task–2491547c40be` · boa-eval-delete-v5 `task–debc1c808576` · ad-attribution-reconciler `task–f8aa9d9efe39` · buddy-allocator-simulator `task–fbbad116c234` · bplus-tree-replay `task–4db5a12cd4c9` · apk-repository-index-audit `task–51ea2184409d` · brew-kettle-boil-ledger `task–63bcba8f1309` · archive-extract-hardlink-symlink `task–337cb9a4735d` · abi-break-variant-ladder `task–3203aa0cf756`. Invalidated/process records: `task–1c31c5d24de1` (material-leak invalidation), `task–edf0073918bf` (routing mis-fire, failed pre-execution). Artifacts are content-addressed (candidate→adversarial→verified→canonical promotion chain journaled per task); frozen-contract SHA-256 preregistration hashes are recorded per task instance. Grader calibrations (stub 0 / oracle 1) are journaled as probe reports per task. Builder/critic costs: tool-reported and token-measured where the transport emits usage; recorded as null where it does not (one lane emits none) — unmeasured values are never fabricated.

Report generated 2026-07-20 at the close of session S24 (sessions S23–S24 executed the batch; the calibration/lifecycle machinery was piloted in an earlier session on a non-corpus task).