

Stepledger: technical report

Every LangGraph node running on Temporal, recorded once per Activity execution in your own Postgres, at any state size.

Measured on macOS-15.7.9-x86_64-i386-64bit; Python 3.11.9, temporalio 1.33.0, langgraph 1.2.12, temporal version 1.9.1 (Server 1.32.0, UI 2.54.1), Postgres 16. Every number is read from bench/results/*.json; each file records the command that produced it.

What beats it

id	attack	measured	expected
7.4	Encrypted payloads	dedupe 11.0x plaintext vs 1.0x encrypted	dedupe is defeated (about 1x); the ledger stays correct; payloads counted as opaque
7.6	Effect whose tool ignores the key; crash before its record	direct: 3 duplicate tickets / 0 needed a person, once_blind: 0 duplicate tickets / 3 needed a person, once: 0 duplicate tickets / 0 needed a person	duplicates only without once(); blind once() stops for a person
7.8	Workflow reset (new run ID)	6 duplicate effects across 3 resets	effects repeat unless the tool's own key dedupes

- With the LLM journal on, 17 billed model calls were still wasted: the worker died between the provider call and the journal write.
- With on_ledger_error="warn", a database outage leaves rows missing until reconcile repairs them from history.
- History still grows, linearly; unbounded runs still need continue-as-new.

Demo 1: the cliff

config	nodes	SDK default	check disabled	largest payload	history MiB
B0 bulk persist at end	36	STUCK persist_all	TERMINATED	2,067,992	39.37
B1 naive per-node write	40	STUCK node 35	TERMINATED	2,067,894	35.42
B2 stepledger (no ext storage)	40	STUCK node 35	TERMINATED	2,067,893	35.43
B3 stepledger + whole-blob storage	40	COMPLETED	COMPLETED	63,873	2.50
B4 stepledger + dedup storage	40	COMPLETED	COMPLETED	63,872	2.50

Demo 2: pull the plug

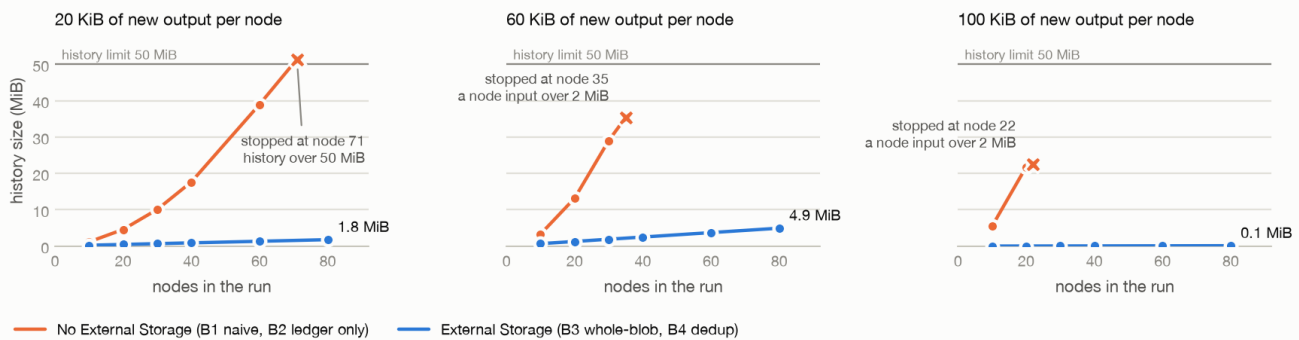
config	runs	faults	dup rows	divergent	lost	orphan	dup effects
B1	20	20	17	13	0	0	7
B1u	20	20	0	5	0	0	4
SL	20	20	0	0	0	0	0

- Stepledger: every declared fault fired once (20), 17 worker restarts, every run completed, each row checked against history.

Demo 3: the quiet quadratic

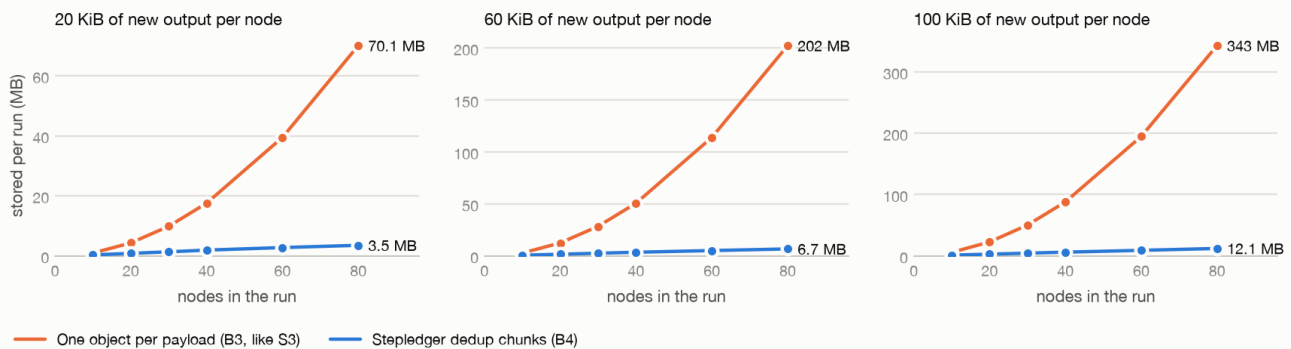
Workflow history size per run

Without External Storage, history grows with the square of the run and the run stops at a wall; with it, history stays small and linear. B1 and B2 match within 0.2%; B3 and B4 histories are identical.



External store bytes per run

Whole-payload objects grow with the square of the run; content-defined chunks stored once grow linearly. Each panel has its own scale.



- At 80 nodes x 100 KiB: 342.97 MB as one object per payload, 12.15 MB as dedup chunks (28.2x).

Demo 4: the view equals the truth

- 100 seeded runs: equal 80, declared gap 20, unequal 0. Cached continue-as-new runs with chain=True: 8 EXACT.

Demo 5: the retry bill

mode	calls billed	USD billed	wasted calls	wasted USD	replays
no-journal	401	4.812	121	1.452	0
journal	297	3.564	17	0.204	79

Overhead

- Ledger write p50 1.842 ms, p95 6.109 ms; 1.387 ms of wall clock per node; history added per node [104.4, 102.75, 101.3] bytes.

What holds under attack

id	attack	measured
7.1	Activity reset	0 divergent rows in 5 reset runs
7.2	Worker clock skew (+/-5 min)	0 divergent rows; a worker-clock fence would keep the stale write for 7 of 18 retried nodes
7.3	Workflow-side nodes	GAP 10/10, wrong EXACT 0
7.5	Non-JSON outputs (pydantic converter, LangChain messages)	0 divergent rows in 15; materialize EXACT 5/5, errors 0

id	attack	measured
7.7	GC retention shorter than the namespace's	refused 1/1; equal retention allowed 1/1