COLLABORATIVE RESEARCH | FRAMEWORKS | YEAR 5 OF 5

ChronoLog

A distributed shared tiered log, ordered by physical time

PI TEAM Anthony Kougkas, Xian-He Sun, Kyle Chard

PROJECT MEMBERS Jaime Cernuda, Eneko Gonzalez, Kun Feng, Inna Brodkin

NSF CSSI 2104013

A **shared log** that captures **activity data** from the edge to HPC, orders it by **physical time** with no central sequencer, and tiers it across memory and storage. This year: **v3.1.0 released**, packaged for Linux, and **roadmap complete**.

```
docker pull ghcr.io/grc-iit/chronolog:latest
```

https://chronolog.dev

01 Abstract

Modern systems generate data at rates exceeding TB/s: sensors, instruments, IoT devices, and human activity across web, mobile, and edge. What they need to capture is **activity data**, events as they occur rather than static state, at a scale existing logging cannot deliver.

ChronoLog is a distributed log store for this regime, from edge to HPC: ordering by physical time, multiple storage tiers, concurrent access, and range queries over partial logs.

02 Key Contributions

Four design principles distinguish ChronoLog from prior log stores:

- **Physical time** — orders events by wall clock, no synchronizations
- **Lock-free append** — avoids slow tail finding caused by protecting locks
- **Auto-tiered** — stores data in best tiers in memory and storage hierarchy
- **Range queries** — retrieves using time range, perfect for time series data

03 Data Model

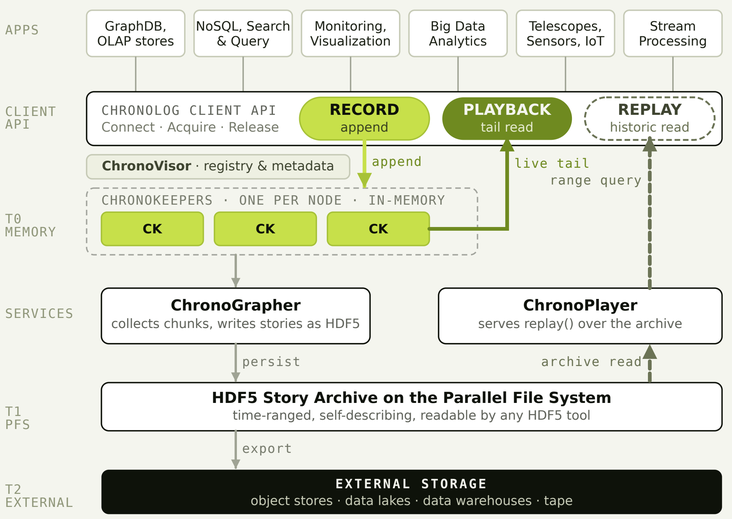
ChronoLog organizes activity data in a three-level hierarchy:

- **Chronicles** partition the namespace, like directories
- **Stories** carry time-ordered streams of data, like files
- **Events** hold individual data records, like lines



04 Architecture & Ecosystem

One time-ordered log, three access paths: **Record** appends into node-local memory, **Playback** reads the live tail from the Keepers, **Replay** reconstructs any past range from the archive.



- **ChronoVisor** — Client and component registry, metadata
- **ChronoKeeper** — In-memory ingestion per node; serves live `playback()`
- **ChronoGrapher** — Builds stories from chunks, persists as HDF5 on PFS
- **ChronoPlayer** — Serves `replay()` over any archived time range

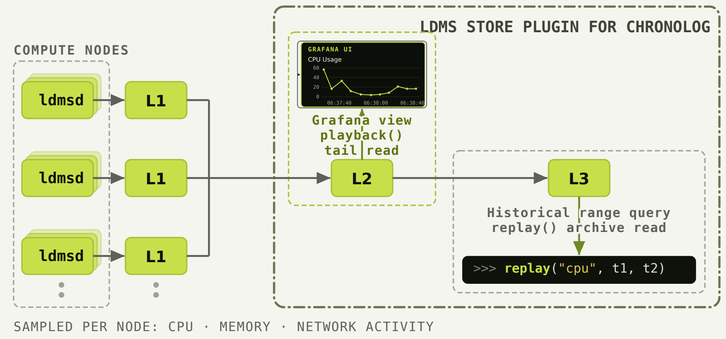
SERVING LAYER · PLUGGABLE INTERFACES OVER THE SAME LOG

Native Log API | **ChronoKVS Key-Value** | **ChronoSQL Query** | **ChronoPubSub Pub/Sub** | **ChronoViz Grafana** | **ChronoLog MCP**

06 Integration

PART A · HPC TELEMETRY — LDMS

Metric streams from the LDMS aggregation tree



`ldmsd` SAMPLER DAEMON ON EVERY COMPUTE NODE L1 L2 L3 AGGREGATOR LEVELS

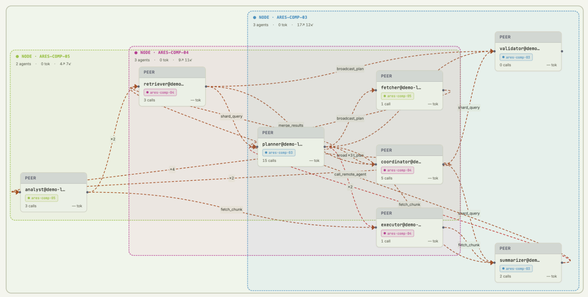
- **ChronoLDMS — an `ldmsd` store plugin.** Aggregators route metric sets to `store_chronolog`: the policy's container becomes a Chronicle, its schema a Story, each sample one JSON event.
- **ChronoViz turns the archive into a live Grafana panel.** Telemetry logged through ChronoLDMS is rendered in Grafana via a native ChronoViz plugin; operators inspect any time range through concurrent range queries served across all tiers, with no sequencer in the path.
- **C glue over a C++ client.** A C-ABI bridge shares one ChronoLog client across `ldmsd` store policies and builds only when OVIS/LDMS headers are present.

LDMS | CHRONOVIZ | AGGREGATOR TREE | RANGE QUERIES | GRAFANA PLUGIN

PART B · DISTRIBUTED WORKLOAD MONITORING — LIVE DEMO

Observability over the shared log

Workers on Ares log every call, hand-off, and error into ChronoLog as timestamped events. Because the log is shared and time-ordered, failures on different hosts are correlated into incidents as they happen, with no separate monitoring database.



LIVE TOPOLOGY · 8 workers on 3 Ares nodes; every inter-node call rebuilt from the log

INCIDENTS fleet-demo + live-demo

- **Failed inter-agent calls: peer timed out** ×78
  7 host(s) · call_remote_agent
  recurring ×175
- **Slow LLM turns** ×51
  8 host(s) recurring ×132
- **Failed inter-agent calls: peer timed out** ×37
  6 host(s) · call_remote_agent
  recurring ×183
- **Rate-limited LLM calls: rate limit exceeded** ×12
  1 host(s) recurring ×60
- **HTTP errors on LLM calls: internal server error** ×8
  1 host(s) recurring ×40

- **Failed inter-agent calls: peer timed out** critical ×78

WHY THIS IS HAPPENING

Calls are exceeding their deadline. At fleet scale this is usually a slow/overloaded peer or transport contention, not a logic bug — the callee can't answer within the window the caller allows.

HOW TO FIX IT

Inspect the slowest to_host in this incident (see the host list) and its queue depth. Raise the call timeout only if the work is legitimately long; otherwise shed load, parallelise the callee, or move the hot agent off a saturated node.

BLAST RADIUS
78 occurrence(s), 7 host(s), 9 agent(s), 2 scenario(s)

CONFIDENCE
70% (heuristic)

AFFECTED HOSTS
ares-comp-050, -01, -05, -02, -06, -03, -04

SCENARIOS
fleet-demo, live-demo

INCIDENTS AND DIAGNOSIS · failures grouped by signature, correlated across hosts, with a suggested fix

07 Performance

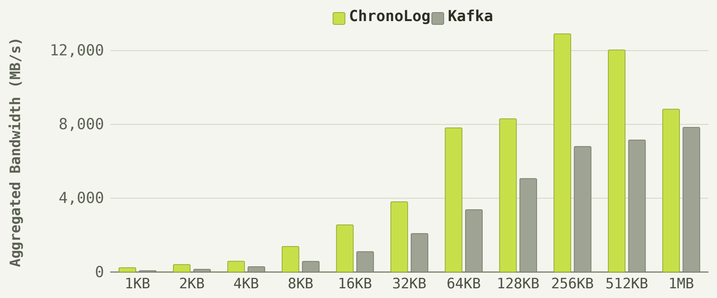
TESTBED

Ares Cluster at Illinois Tech

MACHINE	Ares Cluster at Illinois Tech		
NETWORK	40 Gbps Ethernet	CHRONOKEEPER / BROKER NODES	8
CLIENT NODES	8	CLIENTS PER NODE	20
TOTAL CLIENTS	160		

EVALUATION 1 · RECORD BANDWIDTH

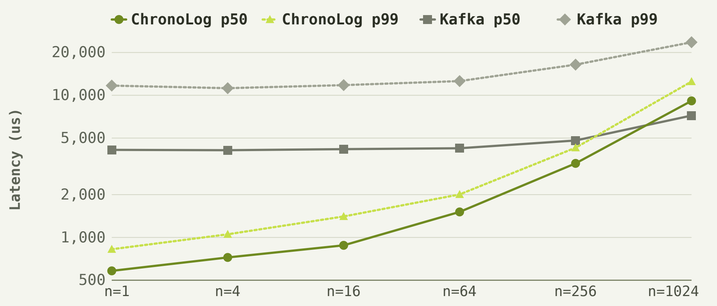
Record Bandwidth



- **What it measures:** how fast ChronoLog can absorb users' data as it arrives, without falling behind the incoming write rate.
- **1.1×–3.2× faster** than Kafka at every record size tested, across the full 1KB–1MB range measured.
- **Peak throughput of 12,905 MB/s** at 256KB records, 1.9× Kafka's 6,803 MB/s at the same size, narrowing to 1.1× by 1MB.

EVALUATION 2 · PLAYBACK

Tail read latency



- **What it measures:** how fast users can read back an event shortly after recording it — the delay between an append and a consumer seeing it.
- **7.1× lower p50, 14.2× lower p99** than Kafka at low queue depth, where ChronoLog's tail read avoids Kafka's consumer-group overhead.
- **Kafka's p50 edges ahead at n=1024** (7,201 vs. 9,135 µs), the one regime tested where ChronoLog's median lead narrows — though its p99 tail stays 1.9× better.

08 ChronoLog Is Available

v3.1.0 · ROADMAP COMPLETE · GET IT

```
# container
docker pull ghcr.io/grc-iit/chronolog:latest
# debian / ubuntu  ·  rhel / fedora / rocky
sudo apt install ./chronolog-linux-x86_64.deb
sudo dnf install ./chronolog-linux-x86_64.rpm
```

- Also a binary tarball, or from source (Spack env + CMake) · C++ SDK, Python bindings · **chronolog.dev/docs**

YEAR 5 BY THE NUMBERS

12	1,204	1,225	11.3k
RELEASES v2.1.1 → v3.1.0	COMMITS MORE THAN Y1–4 COMBINED	CI RUNS ~10 MIN AVG BUILD	REPO CLONES 14× YEARS 1–4

LINKS

chronolog.dev — project site

grc-iit/ChronoLog — source on GitHub

Documentation — quick start, guides

Releases — deb, rpm, tarball

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. **2104013**. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Computing resources: the Ares cluster at Illinois Tech; NCSA Delta and DeltaAI; Chameleon Cloud (NSF).

Related efforts and collaborators: IOWarp (NSF), LDMS / OVIS (Sandia National Laboratories), The HDF Group.

Gnosis Research Center

Illinois Institute of Technology

University of Chicago