Layer1Labs Silicon

LAYER1LABS SILICON INC. · GUIDE

Applying ChronoHive Admission Control to Existing Storage Workflows

Where admission decisions attach in a checkpoint and storage workflow, what a pilot measures, and the limits published alongside the results.

Version 1.0 · October 26, 2026 · Layer1Labs Silicon Inc.

This guide is for a team that already runs training or data workflows with checkpointing, and is considering ChronoHive's admission control for them. It answers three questions in the terms Layer1Labs publishes: where the admission decisions attach in a workflow like yours, what changes and what does not, and what a pilot would measure before you commit to anything.

Everything here is drawn from what Layer1Labs has published about ChronoHive, with each figure carrying the label it was published under — Measured, Demonstrated, or Modeled. Where the published material does not describe a procedure, this guide does not invent one; it tells you what to ask, and what a good answer looks like. Section 6 collects the published limits in full, including the regimes where ChronoHive does not help.

See it live: The product, its measured results, and its published limits — `https://layer1labs.ai/chronohive/`

See it live: This guide in the document catalog — `https://layer1labs.ai/docs/`

© 2026 Layer1Labs Silicon Inc. All rights reserved. See the Notices at the end of this document.

Contents

1 The problem admission control answers **3**

2 Where the decisions attach **3**

3 What applying it does not change **4**

4 The published evidence **4**
4.1 Admission gating, measured 4
4.2 The contention demonstration 5
4.3 Supporting context 5

5 What a pilot measures **5**

6 Prerequisites and limits **6**

7 Questions to ask before you start **6**

Colophon **7**

Notices **8**

1 The problem admission control answers

Training clusters stall expensive graphics processors on storage input and output. Two kinds of work cause most of it. Checkpoint writes are the periodic snapshots that let a training run resume after a failure. Data prefetch is what feeds the next batch to the processors. The two contend for the same storage bandwidth, and nothing arbitrates between them: storage systems offer no application-level admission control — nothing decides which work gets the bandwidth, or when.

The result is idle accelerators, and end-to-end training time dominated by checkpoint stalls on slow or shared storage. There is a second, quieter problem: framework checkpointing carries no end-to-end content verification, so a corrupted checkpoint is typically discovered at restore time — after the run it was meant to protect is already lost.

Admission control is the missing arbitration. Instead of letting checkpoint writes, prefetch, and everything else fight for storage bandwidth as it comes, each piece of work is admitted against real capacity, under a policy that was decided in advance. ChronoHive is a deterministic execution engine that supplies exactly that: the policy is written as a program, compiled once, and then enforced the same way on every run.

2 Where the decisions attach

In a ChronoHive workflow, the admission decision sits between the work and the storage it needs (Figure 1). Three kinds of work pass through it, as published:

- **Checkpoint writes.** Admission is tuned to real storage capacity, so snapshots are written when the storage can take them — fewer stalls, and zero duplicate writes.
- **Data prefetch.** Prefetch and optimizer work are coordinated against the same real capacity, keeping the accelerators fed instead of stalled.
- **The rest of the pipeline.** One policy drives hard admission accounting across graphics-processor memory, storage, and network reservations — capacity is granted deliberately, never by contention.

The policy itself is a compiled artifact, and that is what makes the arrangement practical to adopt. Work is authored as programs in Lingua Franca — a programming language built for deterministic, timed systems, in which programs declare what happens and exactly when. Our compiler, chronoc, compiles the program into a single constraint specification (a .cspec): one self-contained artifact carrying the whole program — its operations, dependencies, capacities, and schedule. The ChronoHive engine executes that specification, enforcing the compiled policy for prefetch, checkpointing, and dispatch deterministically, instead of letting a heuristic scheduler guess at runtime.

The connection points between a program and the outside world are its effectors. In an existing workflow, effectors bind to your storage and compute through software callables: the program asks, the binding does the work, and the admission decision governs when the work is allowed to proceed. Because the policy is compiled rather than improvised, the same program with the same inputs makes the same decisions, bit-for-bit — any run can be replayed exactly, which is what makes an admission policy auditable after the fact.

© 2026 Layer1Labs Silicon Inc.

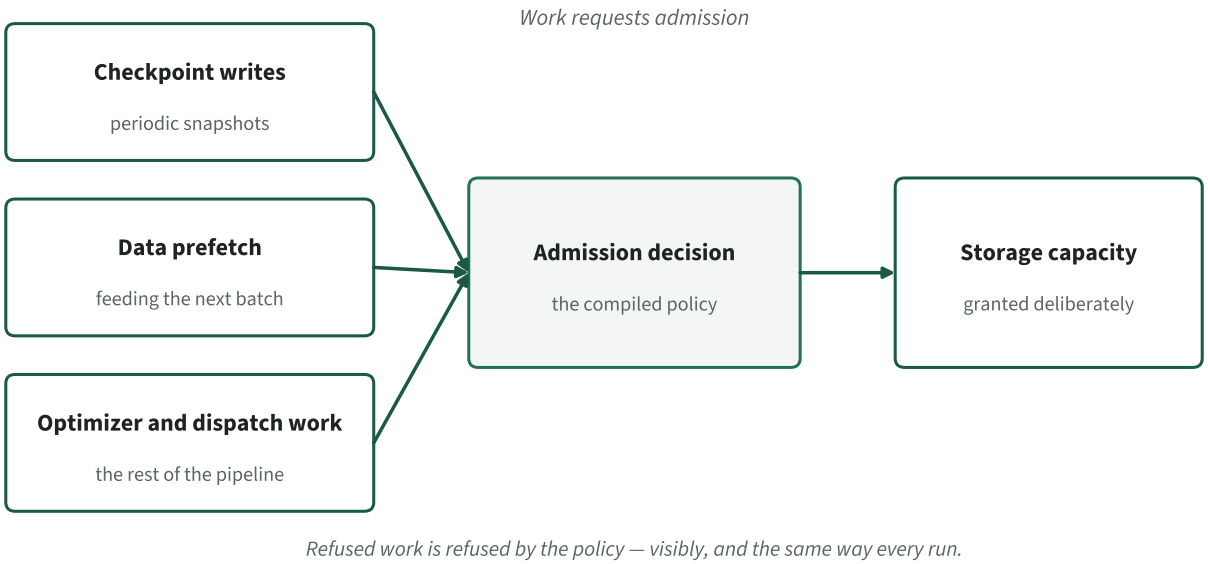


Figure 1: Where admission decisions attach. Checkpoint writes, data prefetch, and the pipeline's other work request admission from the compiled policy; storage capacity is granted deliberately, under that policy, instead of being taken by contention. Source: layer1labs.ai/chronohive/, "What It Does."

3 What applying it does not change

- **Your infrastructure.** ChronoHive runs on the clusters you already own; no hardware purchase is required. You keep your infrastructure, and the technology is licensed into it as software, by written agreement.
- **Your checkpoints' meaning.** Checkpoints are still the snapshots your framework writes — with verification the framework alone does not provide: in the published campaign, 400 of 400 checkpoints were read back and verified, and final training states were bit-identical in 20 of 20 trial groups.
- **Your other options later.** The same programs also run on ChronoFabric hardware when enforcement on the device is wanted; only the effector binding changes. Adopting the software tier does not commit you to the hardware tier, or the reverse.

One thing does change, and it is worth naming: work that used to proceed by contention can now be refused. That is the point of admission control — but it means the policy has to be written to match what your workflow can tolerate, which is what the pilot process in Section 5 is for.

4 The published evidence

Two published results speak directly to admission control. Both carry their labels, and the labels matter.

4.1 Admission gating, measured

With admission control enabled, a preregistered A/B test measured wall-clock improvement on three platforms (Figure 2): +9.85% on an Amazon EC2 g5.xlarge cloud instance (NVIDIA A10G), +7.95% on an RTX 4070

SUPER, and +5.54% on an RTX 5060 Ti. The test ran ten interleaved trials of 200 steps on real data; the FAQ records 22 completed preregistered trials across the platforms.

Every one of those figures sits under the 10% confirmation bar Layer1Labs set for itself. The published verdict, which this guide repeats exactly, is that the admission result is **a positive observation, not a proof**: positive on every platform, and not claimed as more than that.

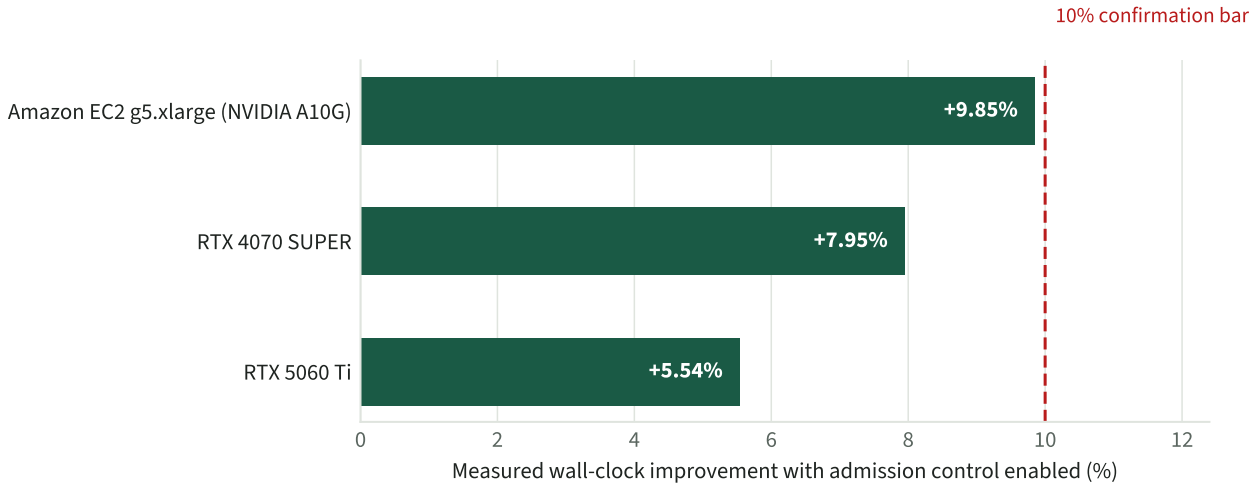


Figure 2: Measured wall-clock improvement with admission control enabled, by platform, against the 10% confirmation bar. Labeled Measured; reported by Layer1Labs as a positive observation, not a proof. Source: layer1labs.ai/chronohive/, "Measured Results"; layer1labs.ai/faq/, "Where does ChronoHive not help."

4.2 The contention demonstration

A second result shows what the policy does when demand exceeds capacity. In the published contention demonstration, labeled Demonstrated, an unmanaged baseline admitted 480 operations and refused none — everything proceeded, and the storage took the contention. The coordinated policy admitted 401 operations, refused 119, retried 40, and dropped 79: the programmed policy, enforced deterministically. Read it as a picture of the mechanism, not as a performance figure: under admission control, refusals are decisions the policy makes on purpose, in the open, the same way on every run.

4.3 Supporting context

The admission results sit inside a wider measured record, stated in full in the companion document *ChronoHive Measured Results*. The parts an evaluator of admission control will want alongside it: determinism measured at 92 of 92 pipeline runs reproduced exactly, with zero duplicate writes; the per-run checkpoint stall in the large-checkpoint campaign cut from 10.5–11.7 seconds to 0.3–1.5 seconds; and the asynchronous checkpoint writer's memory footprint bounded at 3× the checkpoint size, measured across 40 preregistered trials. Layer1Labs also holds a formal proof of the engine's determinism; it is unpublished, and available to qualified partners and investors under NDA.

5 What a pilot measures

A pilot is how an organization finds out what admission control does on its own workflow, in its own storage regimes — because regime decides everything in the published record, and no figure from another team's

hardware answers the question for yours.

As published, pilots are isolated and software-first: no hardware purchase, and your production workflow is not the test bed. Validation is preregistered jointly — the methodology, the arms, and the success criteria are fixed in writing, with Layer1Labs, before any run. That discipline is the same one behind the published figures: Layer1Labs' results come from 501 completed preregistered trials on real graphics processors, with an internal red team recomputing every headline figure before publication. The deliverable of a pilot is a joint benchmark report with reproducible artifacts, and the methodology and trial data behind the published results are available on request.

For planning — and only for planning — Layer1Labs publishes a modeled sensitivity range of 5–15% throughput uplift. It is labeled Modeled, and it is not a production speedup: it is the range the pilot's success criteria should be set against, not a promise of where inside it any given workflow will land. Where a workflow falls depends on its regime: in the published campaigns, the same class of workload measured very differently over slow, shared storage than over fast local NVMe, and the figures from separate campaigns are never stacked or merged.

6 Prerequisites and limits

What you need is modest: an existing cluster (no hardware purchase), workflows whose checkpoint and prefetch behavior can be expressed as programs, and a willingness to fix the pilot's methodology in writing before it runs. The limits, as published, are not modest, and they are part of the offer:

- **Two regimes trail.** CPU-bound workloads on fast NVMe still trail synchronous checkpointing by 6–14% in Layer1Labs' measurements.
- **The hardest measured case.** Very large checkpoints (about 4 GB) written at short intervals (about 3.5 seconds) still trail synchronous checkpointing by about 32%, even in the best configuration — a coalescing writer whose memory stays bounded at 3× the checkpoint size. The limit is encode speed, not memory: encoding the checkpoint takes about 10 seconds against a 3.5-second interval.
- **The admission result is an observation.** The admission-gating figures in Section 4 are positive on every platform and below the publisher's own 10% confirmation bar on each (Section 4). Treat them as a reason to run a pilot, not as a settled gain.
- **The validated scope is small.** Every published measurement is single-node, on at most 2 graphics processors. No multi-node or cluster measurement exists yet; any cluster figure in Layer1Labs' materials is a labeled scenario or model, never a measurement. Multi-node validation from 8 to 32 graphics processors on shared storage is the funded next step.

7 Questions to ask before you start

The published material does not specify every step of applying admission control to a particular storage stack, and a guide should not pretend otherwise. These are the questions the published record equips you to ask — each with the answer, or the kind of answer, Layer1Labs has already put in public:

- **"Show me the admission figures with their basis."** They are in Section 4: measured, preregistered, positive on every platform, under the publisher's own confirmation bar — and the methodology and trial data are available on request. Ask for them.

© 2026 Layer1Labs Silicon Inc.

- **“How do effectors bind to our storage?”** Through software callables, per the published integration account. The specifics for your stack are an evaluation conversation: technical questions go to info@layer1labs.ai.
- **“What will the pilot’s success criteria be?”** Whatever you and Layer1Labs fix in writing before any run — methodology, arms, and success criteria. If a pilot is proposed without that step, it is not the process described here.
- **“Can we see the determinism proof?”** It is unpublished, and available to qualified partners and investors under NDA. The measured determinism record (Section 4) is public.
- **“What does licensing look like?”** ChronoHive is proprietary technology of Layer1Labs Silicon Inc., offered for commercial licensing by written agreement, with an evaluation path for qualified teams: licensing@layer1labs.ai.

Colophon

Sources: the ChronoHive product page and the Frequently Asked Questions page at layer1labs.ai, read in full on October 26, 2026. Every figure in this guide is printed with the basis it was published on and the label it was published under. The modeled range appears once, labeled, in Section 5. Figures were generated by script from a data file transcribed from those pages; no figure value was typed into a figure by hand. This document was prepared with the Layer1Labs document template family and typeset with LuaLaTeX (TeX Live 2026).

© 2026 Layer1Labs Silicon Inc.

Notices

Copyright. © 2026 Layer1Labs Silicon Inc. All rights reserved.

Sharing. This document may be shared in its complete, unmodified form, with attribution to Layer1Labs Silicon Inc. No other reproduction, modification, adaptation, or derivative use — in whole or in part — is permitted without the prior written permission of Layer1Labs Silicon Inc.

No license. Nothing in this document grants, by implication or otherwise, any license or right under any patent, copyright, trademark, or other intellectual property of Layer1Labs Silicon Inc. Publication of this document is a disclosure of information only.

Patents. The technology described in this document is the subject of pending patent applications, including 9 provisional patent applications filed with the U.S. Patent and Trademark Office. No statement in this document is a representation as to the scope, validity, or enforceability of any application or of any patent that may issue from one.

Trademarks. Layer1Labs, ChronoHive, and ChronoFabric are trademarks of Layer1Labs Silicon Inc. Other names mentioned in this document may be trademarks of their respective owners; their use here is for identification only and claims no affiliation or endorsement. In particular, Lingua Franca is an open-source research project of its own community.

Measurements and basis. Figures in this document are drawn from the project's measurement records and are cited where they appear; where a figure is modeled rather than measured, it is labeled as modeled. This document is provided for information only, on an "as is" basis, without warranty of any kind, express or implied, including as to accuracy, completeness, or fitness for a particular purpose. It describes a demonstration system; it is not a specification, an offer, or a forecast of results in any other setting.

Current version. Documents in this family are revised in place under version numbers. The current version of this document is the one published at layer1labs.ai; earlier versions are superseded.