Large-Scale Binary Gibbs Updates on a Commodity GPU

Measured Capacity and Throughput

Deep SIML Labs
siml.life

Version 1.0 · 14 November 2026
Technical capability white paper; not peer reviewed

Abstract

For general readers. A graphics card can hold many simple probabilistic variables, but fitting them in memory and repeatedly updating them are different capabilities. We measure both on one consumer GPU. A specialized SIML implementation sustains updates of about 84.26 billion binary variables and, on a separate matched workload, runs about twelve times faster than an optimized byte-based CUDA control. The advantage depends on workload size: the control wins on the smallest tested ensemble. These are measurements of a defined mathematical model, rather than a simulation of 84.26 billion molecules or independent posterior samples.

Technical summary. We evaluate GPU-resident checkerboard Gibbs updates for a binary, periodic, three-dimensional nearest-neighbor model on one NVIDIA GeForce RTX 3060 with 12 GiB VRAM. Comparators are an optimized UInt8 CUDA implementation, direct JAX GPU code, and upstream THRML sampling software through JAX. Ten paired independent-process rounds, with choices frozen before held-out timing, measure four common populations. At 256 independent 32^3 lattices, SIML achieves a 12.0803× update speedup over the UInt8 control; the pointwise 95% paired-bootstrap interval is $[11.8238, 12.6978]$. Under a common 10,583,146,496-byte device-allocation budget, it sustains 84,258,095,104 cells in one connected lattice at 2.972 full sweeps/s for at least 600 measured seconds. Separate 60-second checks establish populations at 1, 10, and 60 sweeps/s. All 228 declared small-fixture posterior checks pass. These checks and sustained execution do not establish global mixing or effective independent-sample rates at capacity scale. We report operational limits, the small-workload loss, and provenance limitations. Implementation and experimental archives remain private; this PDF reports measured capability without releasing the execution design.

Keywords: binary Gibbs updates; GPU-resident state; sustained capacity; matched benchmarking; probabilistic computation.

1 Question, contribution, and prior work

The practical question is how much binary stochastic state can remain on a commodity GPU while the device continuously updates the entire population. Three quantities matter: time per matched update workload, the resident population sustained over time, and the population sustainable at a requested update rate. They answer different engineering questions. A small population may update quickly while leaving much of the device idle; a large resident population may fit yet update too slowly for a particular application.

This paper reports a completed internal evaluation of those quantities. Its contribution is a bounded capability measurement with explicit correctness gates, comparison boundaries, failure categories, and retained negative results. It makes no priority claim for binary representations, checkerboard scheduling, or GPU Monte Carlo. Weigel studied GPU spin-model simulation on periodic lattices [1]; Fang and colleagues investigated compact multispin methods for a three-dimensional spin glass [2]. Romero and colleagues compared several GPU approaches, including multispin coding, for the two-dimensional Ising model [3]. Those results establish substantial prior work in this area. Their models, update rules, devices, and measurement boundaries differ from this study, so their published rates are not direct baselines for our tables.

The present comparisons use four implementations on the same GPU. The strongest low-level control is UInt8 CUDA, which also uses custom CUDA kernels. Thus an improvement over that control is a comparison between specific CUDA implementations, not an advantage over CUDA as a platform. JAX supplies a compiled-array implementation [4]; THRML supplies an upstream graphical-model sampler [5]. Both run as software on the same GPU. No physical thermodynamic sampling device was measured.

2 Public mathematical target and units

Let $x_i \in \{0,1\}$ denote a cell at site i in an even-sided periodic cubic lattice. Let $\mathcal{E}$ contain each undirected nearest-neighbor edge once. The target distribution at inverse temperature β is

$$\pi(x) = Z^{-1}\exp[-\beta E(x)], \qquad E(x) = \frac{\alpha}{2}\sum_i (x_i - y_i)^2 + \frac{\lambda}{2}\sum_{\{i,j\}\in\mathcal{E}} (x_i - x_j)^2. \tag{1}$$

Here Z is the normalizing constant. A site's color is the parity of the sum of its three coordinates. Observations are $y_i = 0.3$ on even sites and 0.7 on odd sites. All throughput and capacity results use dense support, $\alpha = 1$, $\lambda = 0.25$, and $\beta = 1$. For a site with neighbor set $\mathcal{N}_i$ and degree d_i, the conditional is

$$\Pr(x_i = 1 \mid x_{-i}) = \sigma\left(\beta\left[\alpha(y_i - \tfrac{1}{2}) + \lambda\left(\sum_{j\in\mathcal{N}_i} x_j - \tfrac{d_i}{2}\right)\right]\right), \qquad \sigma(t) = \frac{1}{1+e^{-t}}. \tag{2}$$

A dense site has $d_i = 6$. One *full sweep* updates every even site conditional on the odd state, then every odd site conditional on the newly updated even state. Every performance number counts complete sweeps of the whole reported population.

A *logical cell* is one binary state variable. The ensemble profile grows the number of independent side-32 lattices sharing topology and parameters; its cell count is 32^3C for C chains. The connected profile grows a single periodic lattice and always represents one model. Dividing that connected population by 32^3 describes storage volume, not a number of independent models. Neither profile counts continuous concentrations, molecular species, or effective independent samples.

3 Experimental method

3.1 Platform, controls, and timing boundary

The measured device is one NVIDIA GeForce RTX 3060 with 12 GiB VRAM. Recorded CUDA runtime and driver versions are 13.2; CuPy is 14.2.0. JAX and JAXLIB are pinned to 0.12.0.1. THRML is version 0.1.4 from upstream commit `9c4e6fbb800f5e5c627122e668ff1b158ef3782b`. The four backends implement the same declared energy, population, update schedule, and completed sweep count. SIML and the UInt8 control use specialized GPU implementations; the direct JAX and THRML implementations execute on verified GPU devices.

SIML, UInt8 CUDA, and direct JAX use FP64 setup and 52-bit midpoint Bernoulli decisions. THRML uses its upstream FP64 Ising conditional at $\beta = 1$. Distributional checks establish agreement against the declared fixtures; identical random trajectories across backends are not required. These precision choices and fixture outcomes bound the numerical comparison.

The latency comparison measures resident updates. It includes random generation, both colors, GPU work submission, completion, and synchronization. It excludes initial compilation, model construction, initial state transfer, diagnostics, full-state readback, and artifact serialization. Warm-up occurs before measurement. The state continues evolving across repetitions. These measurements therefore answer a steady-state update question; they do not quantify cold-start or complete application speedup.

3.2 Correctness and held-out comparisons

Correctness precedes timing. The retained experiment index records 39 experiment tests and six harness/provenance tests. Separate statistical checks use exact eight-free-cell conditional posteriors, a within-model joint fixture, and a joint distribution across eight independent chains. Each applicable case uses seeds 0, 1, and 2, 1,024 burn-in sweeps, and 65,536 retained sweeps. The acceptance limits are total variation distance at most 0.05 and maximum marginal error at most 0.02. All 228 declared cases pass. Supplied-randomness unit tests separately check deterministic update behavior, boundaries, and clamps.

Latency choices come from finite training searches using seed 0, then remain frozen before measurement with seed 1. The common workloads are S32/C1, S8/C32, S32/C32, and S32/C256, where S is lattice side and C is the independent-chain count. Each backend receives the same population and work at a given shape. The first two shapes use 2,048 sweeps per repetition; the latter two use 256. Each worker records three timed repetitions. Ten independent process rounds alternate backend order, reducing systematic process-order effects. Separate training at larger populations freezes the capacity choices; a small-population timing choice does not determine capacity by default.

For each geometry, let b_r and s_r be the baseline and SIML worker medians in paired process round r. The reported speedup is

$$R = \frac{\operatorname{median}_r b_r}{\operatorname{median}_r s_r}. \tag{3}$$

Intervals use 20,000 percentile-bootstrap resamples of the ten paired round indices, with bootstrap seed 3060. They are pointwise 95% intervals, rather than simultaneous bounds across all comparisons. The process rounds estimate timing variability; repeatedly timing the same sampling seed does not create independent posterior replications.

3.3 Sustained capacity protocol

All four backend contexts are probed to set a common device-allocation budget of 10,583,146,496 bytes, leaving at least 1 GiB below the minimum observed free memory. Host workers are limited to the lesser of 16 GiB and initially available RAM minus 4 GiB; material worker swapping causes a stop. GPU telemetry is sampled at 1 Hz and worker resident memory at 4 Hz. Compilation has a separate 900-second timeout. Failures retain allocation, host-memory, compilation, and operational-headroom categories.

Population search brackets passing and failing probes where practical. A retained capacity must touch and update the whole state, warm up for 30 seconds, and then complete at least 600 measured seconds of continuous updates with stable allocator boundaries and finite diagnostics. Failure causes a smaller candidate and a new sustained validation. Each 1/10/60-sweeps-per-second frontier receives its own successful check of at least 60 measured seconds. Continuous rates include bounded scalar diagnostics, synchronization, and supervision. Complete trajectories are not retained at these populations.

4 Matched resident-update throughput

Table 1 reports every original comparison. The populations are 32,768 cells for S32/C1, 16,384 for S8/C32, 1,048,576 for S32/C32, and 8,388,608 for S32/C256. A ratio above one favors SIML. All twelve comparisons meet the retained protocol's matching and evidence gates.

Geometry	Baseline	Speedup	Pointwise 95% interval
S32/C1	UInt8 CUDA	1.040	[1.006, 1.077]
S32/C1	Direct JAX GPU	2.294	[2.200, 2.328]
S32/C1	THRML/JAX software	8.516	[8.186, 8.653]
S8/C32	UInt8 CUDA	0.871	[0.832, 0.874]
S8/C32	Direct JAX GPU	2.241	[2.157, 2.300]
S8/C32	THRML/JAX software	7.379	[7.081, 7.500]
S32/C32	UInt8 CUDA	7.756	[7.497, 7.879]
S32/C32	Direct JAX GPU	6.743	[6.607, 6.878]
S32/C32	THRML/JAX software	90.614	[89.354, 91.908]
S32/C256	UInt8 CUDA	12.080	[11.824, 12.698]
S32/C256	Direct JAX GPU	8.810	[8.617, 9.315]
S32/C256	THRML/JAX software	145.720	[142.229, 152.588]

Table 1: Baseline median resident-update time divided by SIML median time; ten paired process rounds per comparison. All intervals are pointwise paired-bootstrap intervals. Both SIML and the UInt8 control use custom CUDA kernels. THRML/JAX is GPU software.

At S32/C256, the baseline and SIML medians are 0.0918164 and 0.00760048 seconds per 256-sweep repetition, respectively. Their ratio is 12.0803×, with interval [11.8238, 12.6978]. This is the principal matched low-level comparison: a substantial difference remains against an explicitly optimized CUDA control implementing the same target. The corresponding ratios against direct JAX and THRML are informative for these adapters, but also reflect their different software abstractions.

The gain changes sharply with geometry and population. At S32/C1 the UInt8 comparison is only 1.040×; at S32/C32 it is 7.756×. At S8/C32 the ratio is 0.871×, and its interval favors UInt8 CUDA.

Accordingly, the results support a size-dependent throughput advantage on three declared shapes, with a retained loss on the fourth. They do not establish a universal advantage for SIML or for any general implementation family.

The S32/C256 result is also a resident measurement. An application that repeatedly changes parameters, constructs graphs, transfers full states, or requires large observation histories incurs costs outside this table. A twelvefold update improvement does not establish a twelvefold improvement for such an application's total runtime.

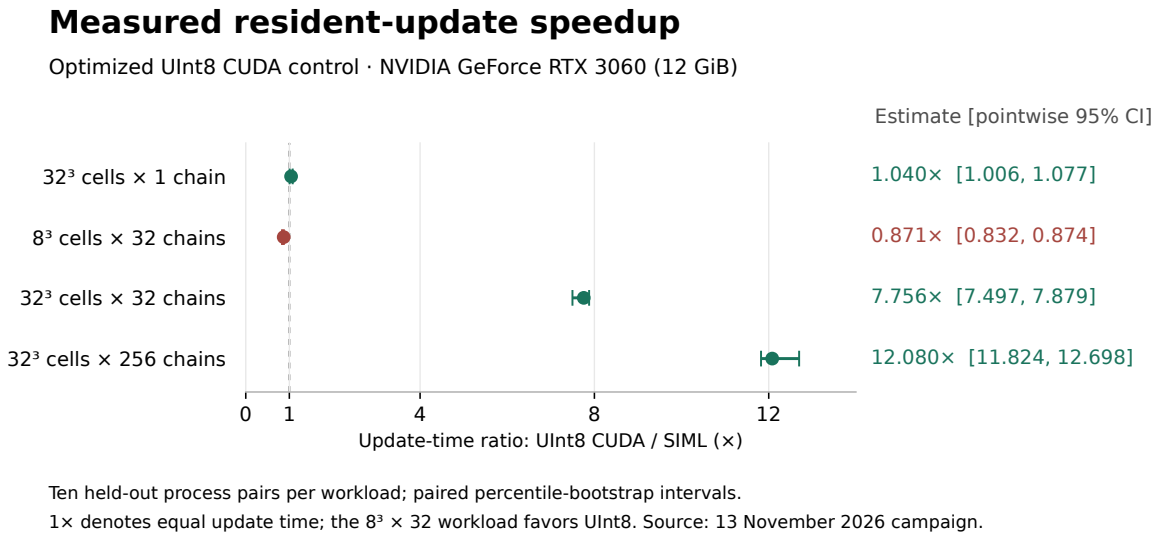


Figure 1: Matched resident-update speedups from the frozen campaign. The UInt8 control wins at S8/C32. Intervals are pointwise 95% paired-bootstrap intervals across ten process rounds. Throughput comparisons use matched logical populations and sweep counts, with compilation and diagnostics outside the timed boundary. Source: experiments 012–014, `resident-20261113-final`, 13 November 2026.

4.1 Separate exploratory tiny-workload follow-up

The main S8/C32 comparison applies the ensemble choice trained at S32/C32 to a smaller population. After observing the loss, a separate exploratory study trains specifically at S8/C32 using seed 0, freezes new choices, and measures seed 2 in ten paired independent-process rounds. This follow-up addresses whether shape-specific training removes the observed disadvantage. Its protocol passes, but its result continues to favor UInt8 CUDA (Table 2).

Exploratory S8/C32 baseline	Speedup	Separate 95% interval
UInt8 CUDA	0.796	[0.795, 0.820]
Direct JAX GPU	2.289	[2.274, 2.670]
THRML/JAX software	7.342	[7.250, 7.434]

Table 2: Separate shape-specific follow-up using held-out seed 2 and 2,048 full sweeps per repetition. These exploratory intervals do not replace or extend the twelve original comparisons in Table 1.

The tiny-workload result is useful for selecting an execution strategy. It demonstrates that fitting the model and preserving its conditional distribution do not determine which backend has the lowest

latency. Population and topology must remain part of any practical performance qualification. No single crossover threshold is inferred from four geometries.

5 Continuous resident capacity

Table 3 gives the eight sustained certifications. Every row represents a separate successful run of at least 600 measured seconds after warm-up. Counts are demonstrated populations under the shared allocation and operational budgets. The word *certification* denotes this internal protocol check; it does not imply independent accreditation or a physical device maximum.

Backend	Population	Logical cells	Models	Sweeps/s
SIML	Ensemble	84,166,049,792	2,568,544	3.125
SIML	Connected	84,258,095,104	1	2.972
UInt8 CUDA	Ensemble	9,992,929,280	304,960	2.043
UInt8 CUDA	Connected	10,532,261,888	1	1.783
Direct JAX	Ensemble	4,294,967,296	131,072	7.494
Direct JAX	Connected	4,283,098,624	1	6.727
THRML/JAX	Ensemble	4,080,009,216	124,512	0.468
THRML/JAX	Connected	7,077,888	1	266.173

Table 3: Demonstrated resident populations on one RTX 3060 with 12 GiB VRAM; each row passed at least 600 measured seconds. Ensemble models share topology and parameters. Connected THRML/JAX capacity is limited by host graph construction under the declared budget. Rows have different cell counts, so their sweeps/s are not matched-work speedups.

SIML sustains one side-4384 connected lattice, totaling 84,258,095,104 binary cells, at 2.9716 full sweeps/s for 600.018 measured seconds. Its ensemble run sustains 2,568,544 side-32 models, totaling 84,166,049,792 cells, at 3.1247 sweeps/s for 600.061 measured seconds. Allocator-reserved bytes are unchanged across the measured boundaries in both runs. These populations demonstrate resident updating rather than allocation without computation.

The connected SIML and UInt8 populations differ by approximately 8.00×; the ensemble populations differ by approximately 8.42×. These compare operational capacity outcomes, not matched runtime. In particular, faster sweeps/s at a smaller population cannot be treated as a faster update implementation without accounting for the work per sweep. The direct JAX row illustrates this distinction: its certified population is smaller while its sweep frequency is higher.

Backend limits also differ. Retained larger-candidate failures include allocation-budget or allocation failure, operational headroom, and host-memory limits. The connected THRML case reaches side 192, or 7,077,888 cells, before host graph construction reaches the declared limit. It does not exhaust GPU state storage. The connected SIML and UInt8 searches end without an observed next-fit failure; their populations are lower bounds rather than bounded estimates of a physical maximum. No claim of globally optimal capacity follows from a finite search.

Capacity-scale observation is intentionally bounded. Scalar diagnostics are collected at most once per second after completed sweeps, with only bounded selected-chain observations. Their overhead is included in continuous throughput. Retaining every state after every sweep, or replacing shared parameters with arbitrary per-site parameters, would introduce a different storage and execution workload.

5.1 Population at requested update rates

Each entry in Table 4 has a separate successful check lasting at least 60 measured seconds. The requested frequency refers to full-population sweeps per wall-clock second. It is not a hard deadline for every individual sweep, and the rows need not use the same population as the 600-second checks.

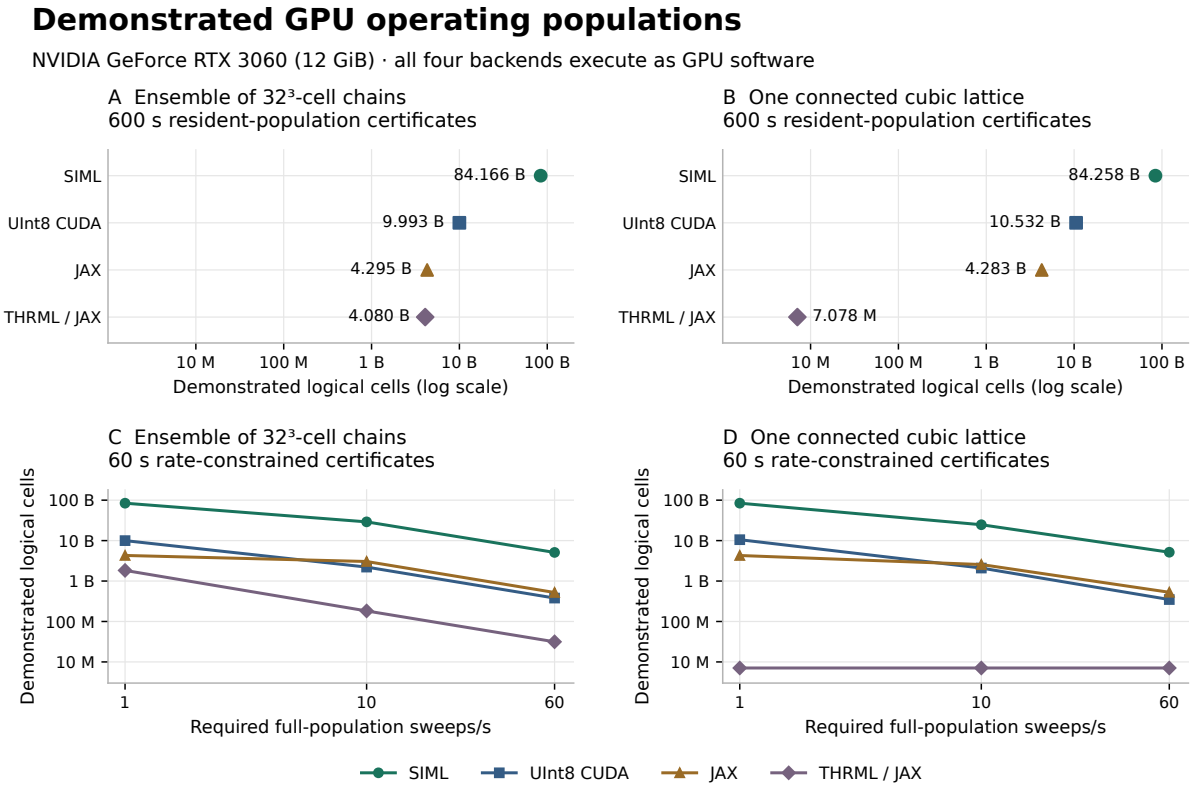


Figure 2: Demonstrated resident populations and rate frontiers for ensembles and one connected lattice. The 600-second capacity and 60-second rate checks are distinct certifications. Logical cells are binary state variables; connected-lattice cell counts do not count independent models. THRML/JAX is software, with its connected profile limited by host graph construction. Source: experiments 012–014, `resident-capacity-20261113-final`, 13 November 2026.

For example, the SIML connected 60 Hz population is 5,159,780,352 cells. That run achieves 60.616 sweeps/s over 60.050 measured seconds. The corresponding ensemble population is 5,100,273,664 cells, achieving 62.865 sweeps/s over 60.033 seconds. These are update rates for the specified model; a 60 Hz requirement for an application would additionally need to include its observation, input, and response costs.

Backend	Population	Cells at ≥ 1 Hz	Cells at ≥ 10 Hz	Cells at ≥ 60 Hz
SIML	Ensemble	84,166,049,792	29,070,721,024	5,100,273,664
SIML	Connected	84,258,095,104	24,693,014,528	5,159,780,352
UInt8 CUDA	Ensemble	9,992,929,280	2,214,592,512	379,584,512
UInt8 CUDA	Connected	10,532,261,888	2,097,152,000	348,913,664
Direct JAX	Ensemble	4,294,967,296	3,044,016,128	521,142,272
Direct JAX	Connected	4,283,098,624	2,560,108,032	527,514,112
THRML/JAX	Ensemble	1,832,910,848	182,452,224	31,457,280
THRML/JAX	Connected	7,077,888	7,077,888	7,077,888

Table 4: All 24 separately certified rate-frontier populations. Hz denotes full sweeps/s. Non-overlapping 60-second window rates use interpolated completed batch counts and are estimates, rather than per-sweep deadline guarantees.

6 Interpretation and limitations

6.1 What the results establish

The campaign demonstrates two capabilities on a consumer GPU: high sustained binary state capacity and substantial resident-update throughput at suitable populations. Their combination can inform engineering of large regular stochastic models. The measured full sweeps change evolving state and preserve the declared update schedule; the experiment does not replay an immutable operator input. Representative retained transfer captures for SIML and the UInt8 control each cover 64 complete sweeps and one diagnostic, recording a single 40-byte device-to-host copy and no host-to-device copies. This supports the stated resident boundary for those captures.

The original comparisons and exploratory loss also define limits to the performance claim. Benefits are substantial for the two larger ensembles and small for one side-32 model; the UInt8 control wins for the side-8 ensemble. Results against THRML reflect both its actual sampler and the costs imposed by a more general graph-based software path. They do not measure a thermodynamic chip, and they cannot establish how such a chip would compare in latency or energy.

The public target and method describe what was measured without specifying how the specialized execution engine implements it. The result is an engineering capability report. Algorithmic novelty, reproducible implementation performance, and broader application utility require different evidence. In particular, a direct comparison with established optimized multispin implementations remains future work.

6.2 Statistical and application limits

The 228 posterior checks enumerate small conditional fixtures; they do not enumerate the giant populations. Long-running finite diagnostics establish that the tested execution remains operational, but cannot certify global equilibrium or convergence of a non-enumerable lattice. Full sweeps are generally correlated. This paper therefore reports no capacity-scale effective sample size, independent samples/s, convergence acceleration, or scientific estimation accuracy. More challenging parameter regimes and multiple independent statistical replications would be necessary for such claims.

The reported workload has binary variables, regular periodic topology, uniform coupling, and two alternating observation values. All capacity measurements use one fixed parameter setting. The ensemble shares topology and parameters across chains. General weighted graphs, heterogeneous

fields, additional state variables, or sample-history retention may change both memory use and runtime. A biochemical reaction model would also need its own state semantics, conservation laws, physical rates, and validation. No molecular kinetic trajectory or chemically realistic simulation is established here.

All performance measurements come from one GPU and software environment. Finite training choices do not establish an optimum over all CUDA or JAX implementations. The pointwise bootstrap intervals describe the retained timing protocol, not portability to another machine or simultaneous confidence across every result. Complete model construction, compilation, transfer, and application observation costs remain outside the matched update timing.

6.3 Instrumentation and provenance limits

Nsight Systems traces are diagnostic and their perturbed durations are excluded from latency statistics. SM/DRAM hardware counters are unavailable under the recorded driver permissions. Temperatures, clocks, power, and throttle observations do not establish occupancy, bandwidth saturation, or a causal explanation of the speedup. No controlled energy-per-update or energy-per-independent-sample comparison is claimed. Allocator, live-array, compiler workspace, and total device-memory boundaries must remain distinct.

Of the 228 retained posterior rows, 216 predate complete host/common-module source inventories. Selected legacy coverage is authorized through recorded kernel or module digests, including 48 selected rows. The complete original host snapshots for those legacy rows were not recorded. Later source archives identify measured files matching recorded identities separately from additional files copied after execution. Hashes support detection of drift against those identities; they are not external signatures or proof of measurement authenticity. This limitation remains part of the evidence record.

7 Evidence availability and next validation steps

This release consists of the reviewed PDF. Manuscript sources, implementation, experimental records, tuning selections, compiler outputs, and detailed archives remain private. The paper is self-contained about its target, units, protocol, reported values, and limitations, but it does not provide a public implementation or independently reproducible benchmark package. No claim of independent external replication is made.

The frozen internal measurement identifiers are `resident-20261113-final` for layout/latency evidence and `resident-capacity-20261113-final` for sustained capacity. The completed capacity campaign ended on 13 November 2026. These identifiers distinguish this evaluation from earlier development experiments; results from those earlier campaigns have not been substituted into the tables here.

Internal evidence retains paired process timings, successful and failed candidate records, correctness traces, source identities, telemetry, and transfer audits. The complete retained artifact inventory contains 4,767 files. A separate read-only check of the selected paper evidence verifies 67 files and 72 exact claim extracts with the required local archives present. That check tests consistency against its manifest, rather than rerunning benchmarks or independently authenticating their origin. The broader internal analysis and original worker records remain the authority for the completed campaign.

The next useful validation steps are replication on another GPU, comparison with established optimized spin-model implementations at matched targets, and parameter sweeps with convergence and

effective-sample diagnostics. Application claims should then be tested with complete input, observation, and response boundaries. These are future studies, rather than prerequisites retroactively assumed to have passed for the present capability measurements.

Measurement scope. The principal result is 12.0803× matched resident-update throughput against the declared UInt8 CUDA control at S32/C256. The sustained connected-population result is 84,258,095,104 binary cells at 2.972 full sweeps/s on the stated RTX 3060. Each retains its own workload, timing boundary, and validation evidence.

References

[1] M. Weigel (2011). *Simulating spin models on GPU*. Computer Physics Communications 182, 1833. doi:10.1016/j.cpc.2010.10.031. Author manuscript: arXiv:1006.3865.

[2] Y. Fang, S. Feng, K.-M. Tam, Z. Yun, J. Moreno, J. Ramanujam, and M. Jarrell (2014). *Parallel Tempering Simulation of the three-dimensional Edwards–Anderson Model with Compact Asynchronous Multispin Coding on GPU*. doi:10.1016/j.cpc.2014.05.020. Author manuscript: arXiv:1311.5582.

[3] J. Romero, M. Bisson, M. Fatica, and M. Bernaschi (2019). *A Performance Study of the 2D Ising Model on GPUs*. arXiv:1906.06297. Related journal article: doi:10.1016/j.cpc.2020.107473.

[4] JAX developers. *JAX: composable transformations of Python+NumPy programs*. Official software repository: github.com/jax-ml/jax. Measured JAX/JAXLIB version: 0.12.0.1. Accessed 13 November 2026.

[5] Extropic. *THRML: Thermodynamic Hypergraphical Model Library in JAX*. Official software repository: github.com/extropic-ai/thrml. Measured version 0.1.4; upstream commit `9c4e6fbb800f5e5c627122e668ff1b158ef3782b`. Accessed 13 November 2026.