Bigger Networks, More Trajectories: Scaling Exact Stochastic Simulation on GPUs through Sparse Storage and a Warp-ary Propensity Tree

Alfia Parvez[1,3], Zan Luthey-Schulten[1,2,3]

NSF STC for Quantitative Cell Biology[1]; Department of Chemistry and Physics, University of Illinois Urbana Champaign (UIUC)[2]; Beckman Institute, University of Illinois Urbana Champaign (UIUC) [3]

I. Introduction

Low-copy-number biochemistry is intrinsically stochastic. Deterministic rate equations report only mean behavior, whereas the Chemical Master Equation (CME) must be sampled with Gillespie's Direct Stochastic Simulation Algorithm (SSA) [1] to recover fluctuations, distributions, and rare-event statistics. Two trends strain the exact SSA on hardware. First, rule-based models routinely expand to tens of thousands of reactions (M). Second, because a single trajectory is one noisy sample, reliable statistics demand thousands of independent replicates. Exact SSA must therefore scale in two directions at once: network size and ensemble breadth, precisely the regime where conventional implementations break. Two walls appear. The memory wall: dense species(N) × reaction(M) stoichiometry and dependency tables reach gigabytes, so large models exceed device memory. The selection wall: Naive Direct-Method reaction selection costs O(M) per step, and each step is a tightly coupled, microsecond-scale dependency that resists parallelization.

II. Background

Exact SSA on CPUs is mature. Ordering methods (Optimized Direct Method [2], Sorting Direct Method [3]) retain O(M) worst-case selection but move frequently firing reactions earlier to cut average cost. Logarithmic methods (LDM [4], Next Reaction Method [5]) reach O(log M) selection via partial-sum trees or indexed priority queues. Composition–rejection [6] achieves expected O(1). Dependency graphs [5] update only the $D \ll M$ affected reactions per step.

GPUs, by contrast, accelerate SSA mainly through coarse ensembles that map one trajectory to one thread; libraries by Köster (2022) [7], and crn-jax [8] launch thousands of trajectories for smaller models. Yet each thread still runs the serial step, so per-step cost and per-trajectory memory grow with dense model size.

Fine-grained GPU schemes parallelize work within each trajectory: Komarov and D'Souza's GPU-ODM [9] parallelizes hierarchical reaction selection and dependency-based propensity updates, whereas Sumiyoshi et al. [10] parallelize propensity evaluation and species updates but retain serial reaction selection. GPU-ODM uses a flattened sparse dependency graph with duplicated reaction metadata, while Sumiyoshi et al. use Compressed Sparse Row (CSR) for the sparse stoichiometric matrix. We found no publicly accessible source release associated with either implementation.

III. Approach

We took a novel approach by building a warp-ary tree inspired by Komarov and D'Souza's search algorithm and combining it with a Compressed Sparse Row (CSR) representation. We introduce an exact GPU-based Gillespie Direct solver into the Lattice Microbes (LM) software [11] that addresses both limitations within a single design. The reaction network, defined by stoichiometry and the reaction-to-reaction dependency graph, is stored once in CSR format, reducing the memory footprint from dense $O(M \cdot N)$ to $O(M + nnz)$. This shared, read-only model remains resident on the device even for the largest reaction networks. The storage layout is shown in Fig. 1.Each trajectory is assigned to a single warp of 32 threads, while thousands of warps execute concurrently, combining fine-grained parallelism within each trajectory with coarse-grained parallelism across trajectories.

We construct a complete 32-ary propensity tree in shared memory. The leaves represent the per-reaction propensities, as shown in Fig. 2, while each internal node stores the sum of its 32 children, with the root representing the total propensity. At each level, a warp-wide ballot and prefix scan are used to select the appropriate branch, allowing a reaction to be selected in $O(\log_{32} M)$ time by descending through $\log_{32} M$ levels. This approach generalizes Komarov and D'Souza's fixed-depth, two-level search [9] into a recursive, warp-width tree structure and replaces its O(M/32) within-block linear scan.

After a reaction fires, only the affected leaf delta is propagated to the distributed level-1 bins using atomic operations, while the small upper levels of the tree are rebuilt. This ensures that the root remains consistent with the current total propensity.

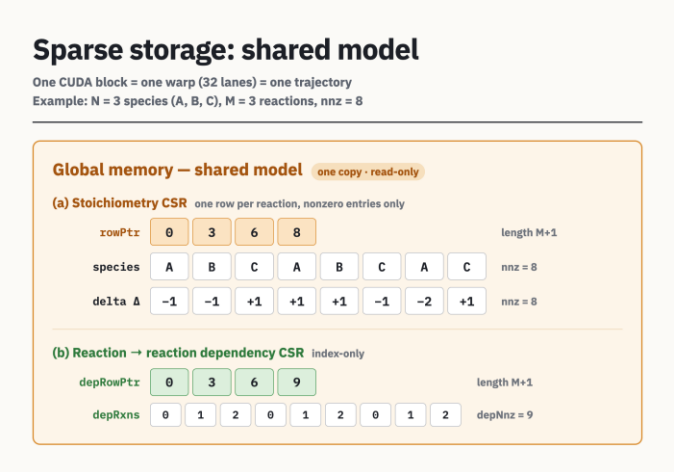


Fig. 1. Sparse CSR storage. The shared read-only model (stoichiometry + dependency graph) is stored once in O(M+nnz); only per-trajectory counts and propensities replicate ×K, never dense O(M·N).

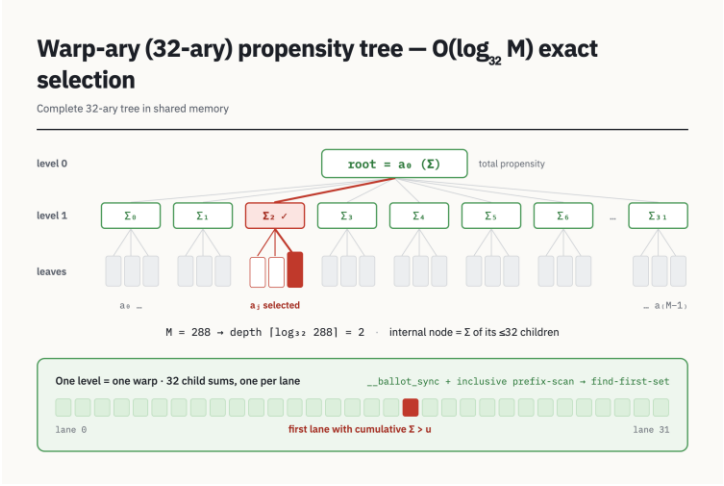


Fig. 2. Complete 32-ary warp-tree. Leaves are per-reaction propensities; each internal node sums its ≤32 children; the root is the total. Selection descends O(log32M) levels, one warp-wide ballot/prefix-scan per level.

IV. Results

On an NVIDIA B200 we benchmarked five models spanning three orders of magnitude: Multistate (M=18), Multisite2 (M=288), EGFR (M=3,749), BCR (M=24,388), and FcεRI γ2 (M=58,276), drawn from the BioNetGen model suite [12], against Köster's CudaDirectMethod [7] and crn-jax [8]. The CSR footprint cuts model memory by up to ~800× on FcεRI (shown in Fig. 3), and our solver completes all five networks without running out of memory. At K=16,384 the dense tools cannot: Köster is infeasible for M ≥ 3,749, and crn-jax exhausts device memory (~0.5 TB dense trajectory store). Logarithmic selection breaks the selection wall, per-step cost stays sublinear ($T_{up} \approx 4.3\times10^{-8} \rightarrow 4.0\times10^{-7}$ s across a ~3,000× increase in M) while coarse Direct-Method tools scale ~O(M); where tools overlap we observe 2–3× to ~12–20× advantages. Ensemble throughput rises with number of replicates and saturates the GPU. Correctness is verified against the CPU reference StochKit2 [13], with ensemble means and count distributions matching within stochastic error. Results are summarized in Fig. 3 to 5.

	MODEL SIZE		NONZEROS ACTUALLY STORED				RESIDENT MODEL MEMORY		
Model	Species N	Reactions M	Dense pairs M × N	Stoich. nnz	Depend. nnz	Fill	Dense	CSR	Saving
EGFR	356	3,749	1,334,644	10,498	5,249	0.79%	16.2 MB	211 KB	77×
BCR	1,122	24,280	27,242,160	52,148	42,283	0.19%	328 MB	1.27 MB	258×
FcεRI γ2	3,744	58,276	191,453,184	133,148	70,144	0.070%	2.30 GB	2.79 MB	824×

Dense baseline: three M × N tables at 4 B, i.e. 12 B per reaction–species pair. **CSR:** 8 B per stoichiometry nonzero (index + coefficient), 4 B per dependency nonzero, plus O(M) row offsets.

Fig. 3. Results: Reduction in memory footprint of models using CSR format

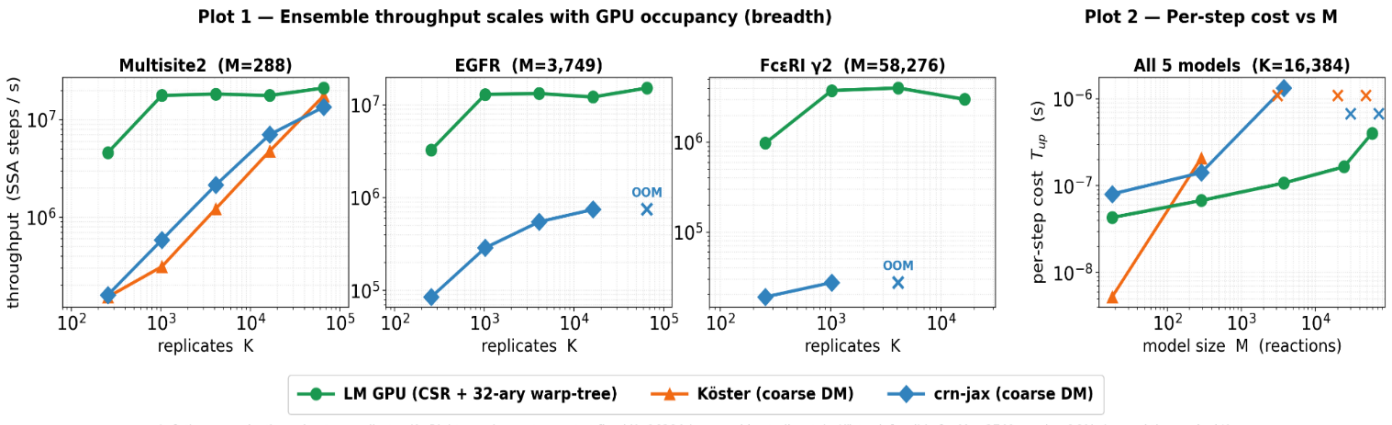


Fig. 4. Plot1: Ensemble throughput vs replicates K. Our solver scales and saturates the GPU; coarse tools trail or exhaust memory. Plot 2: Per-step cost T_up vs model size M at K=16,384. Our solver (green) grows ~10× across a ~3,000× increase in M, far slower than the ~O(M) of coarse tools, which also hit OOM/infeasibility (×).

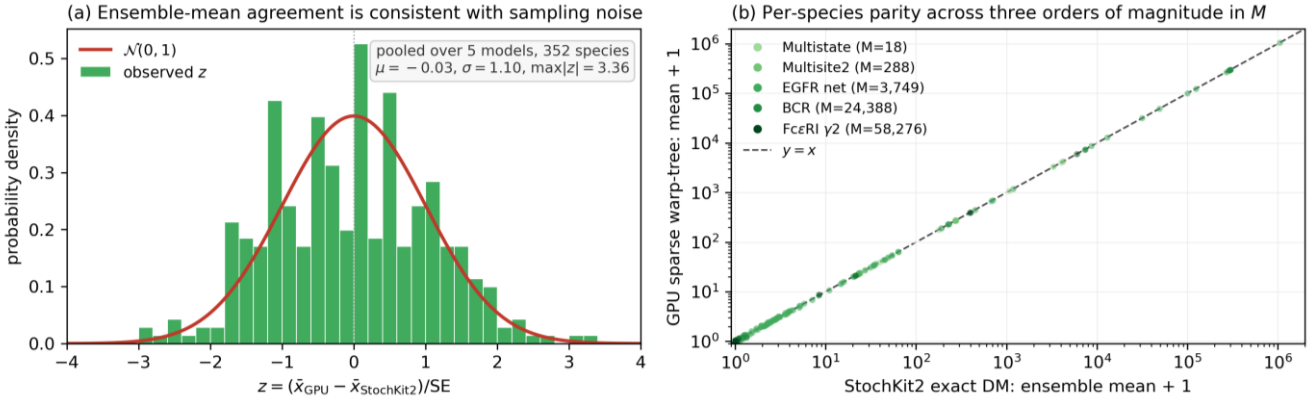


Fig. 5. Validation against StochKit2 exact Direct Method [13]. Pooled per-species standardized mean differences follow N(0,1), as required if both samplers are exact. K=4,000/2,000/400/32/24 at T=10/4/1/0.01/0.02 s.

V. Status and Future Work

The solver is implemented and validated. We plan an open-source release in Lattice Microbes v3.0, with the 32-ary warp-tree as the default GPU selection path, direct SBML [14] and/or BioNetGen [12] import, and the reproducibility benchmark suite. The same engine will accelerate the CME component of whole-cell simulations built on LM.

References

[1] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," J. Phys. Chem., vol. 81, no. 25, pp. 2340–2361, 1977.

[2] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," J. Chem. Phys., vol. 121, no. 9, pp. 4059–4067, 2004.

[3] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova, "The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior," Comput. Biol. Chem., vol. 30, no. 1, pp. 39–49, 2006.

[4] H. Li and L. R. Petzold, "Logarithmic Direct Method for discrete stochastic simulation of chemically reacting systems," Tech. Rep., Dept. of Computer Science, Univ. of California, Santa Barbara, 2006.

[5] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," J. Phys. Chem. A, vol. 104, no. 9, pp. 1876–1889, 2000.

[6] A. Slepoy, A. P. Thompson, and S. J. Plimpton, "A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks," J. Chem. Phys., vol. 128, no. 20, p. 205101, 2008.

[7] T. Köster, L. Herrmann, P. Andelfinger, and A. M. Uhrmacher, "GPU-accelerated simulation ensembles of stochastic reaction networks," in Proc. Winter Simulation Conf. (WSC), 2022, pp. 2570–2581.

[8] R. Henry, "crn-jax: Chemical reaction networks in JAX—a GPU-optimized Gillespie/SSA library," v0.3.1, 2026. [Online]. Available: https://github.com/robinhenry/crn-jax

[9] I. Komarov and R. M. D'Souza, "Accelerating the Gillespie exact stochastic simulation algorithm using hybrid parallel execution on graphics processing units," PLoS ONE, vol. 7, no. 11, p. e46693, 2012.

[10] K. Sumiyoshi, K. Hirata, N. Hiroi, and A. Funahashi, "Acceleration of discrete stochastic biochemical simulation using GPGPU," Front. Physiol., vol. 6, art. 42, 2015.

[11] E. Roberts, J. E. Stone, and Z. Luthey-Schulten, "Lattice Microbes: High-performance stochastic simulation method for the reaction-diffusion master equation," J. Comput. Chem., vol. 34, no. 3, pp. 245–255, 2013.

[12] L. A. Harris, J. S. Hogg, J. J. Tapia, J. A. P. Sekar, S. Gupta, I. Korsunsky, A. Arora, D. Barua, R. P. Sheehan, and J. R. Faeder, "BioNetGen 2.2: advances in rule-based modeling," Bioinformatics, vol. 32, no. 21, pp. 3366–3368, 2016.

[13] K. R. Sanft, S. Wu, M. Roh, J. Fu, R. K. Lim, and L. R. Petzold, "StochKit2: software for discrete stochastic simulation of biochemical systems with events," Bioinformatics, vol. 27, no. 17, pp. 2457–2458, 2011.

[14] M. Hucka, A. Finney, H. M. Sauro, et al., "The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models," Bioinformatics, vol. 19, no. 4, pp. 524–531, 2003.