triton_benchmarks

Snapshot of your Triton benchmarks

Kernel	Work-load (1024 paths × 1 000 steps)	Avg launch time	Speed-up vs. CURAND
fused_sde_stub (RNG + cumsum only)	100 repeats	**25.97 µs**	**≈ 19 ×**
fused_sde_control (full feedback / decay logic)	100 repeats	**30.2 µs**	**≈ 16 ×**
Baseline CURAND C++ kernel	100 repeats	**0.496 ms**	—

That's ~ 29 ns per path-step for the control kernel on an A100 40 GB (Torch 2.7.1, Triton 3.3.1) .

What those numbers mean in practice

1. **GPU saturation is already solid** – a single A100 can crank ~33 M path-steps / ms.

   One million 10 k-step paths would arrive in ≈ 300 ms, well below a sub-second inference budget.

2. **Logic overhead is tiny** – adding your stochastic-control math costs ~4 µs (≈ 14 %) over the bare RNG+cumsum. That confirms the extra branching and tl.where statements aren't a bottleneck yet.

3. **Why Triton beats CURAND** – the fused kernel keeps RNG, maths and stores in-register; the CURAND version burns cycles on separate memory passes and kernel-launch latency.

Quick validation against BICEP + new risk dial

Pipeline stage	Latency slice	Budget after Triton win

Triton SDE + ENN cell	**≤ 0.04 ms**	✔ leaves ~20 ms for GNN-limit dial, Fusion α and router
GNN encoder (PyG on 1024-node contradiction graph)	~1.2 ms	still fits
Fusion α + order routing	~17 ms*	end-to-end ≈ 19 ms

- Measured in earlier Colab runs on same GPU.

So even with the new **limit/colimit micro-risk dial** replacing the single Ising score, you remain safely under the 25 ms end-to-end target.

Head-room & micro-optimisations

Idea	Expected gain	Effort
Counter-based Philox RNG (tl.philox) to eliminate per-step seed math	1-2 µs	2 lines
Shared-memory scratch for acc followed by one final global store	2-3 µs on A100	6 loc
Grid-strided loop (tl.program_id(axis=0) + …) to process ≥ 2 paths / block	Keeps occupancy high when n_paths < SM count	<10 loc
Autotune block size via Triton's meta['autotune']	+5-8 % throughput across GPUs	drop-in decorator

These refinements would push the control kernel toward **~22–24 µs** for the same workload – but they're nice-to-have rather than required.

Real-world impact for trading

- **Intraday Monte-Carlo refresh:** you could regenerate a **100 k-path, 5 k-step** Brownian cloud (~3.5 GB of output) in ≈ 15 ms, enabling path-level uncertainty updates every price tick without stalling the book.
- **Risk dial synergy:** the ultra-low SDE latency lets your new **limit/colimit micro-risk gauge** query fresh path statistics (vol-of-vol, excursion probabilities) in the same window it evaluates spread and depth.

- **Edge in options:** For vol surfaces you can afford to Monte-Carlo Greeks on-demand rather than rely on cached grids – handy when contradiction shocks blow out implied vol.

Bottom line

Your Triton kernels already deliver a **16–19× speed-up** over the CURAND baseline, shaving the stochastic-sampling slice down to **<0.05 ms**. That keeps the full Fusion α + BICEP + limit-dial stack comfortably inside a real-time (<25 ms) trading loop, with a bit of head-room left for future bells and whistles.