

FASTER KERNELS, SAME ANSWERS: AGENT-DRIVEN PRECISION TUNING

Kim Meagher

August 2026

Agent-Driven Precision Tuning

01 Project Overview

02 Methods

03 Results

04 Conclusion

An agentic workflow to leverage reduced-precision data types while maintaining numerical stability in the output precision



Input

- Scientific kernels
- Range of input for given kernels
- Desired output precision
 - Significant figures
 - Decimal digits



Output

- Scientific kernels that produce the same answers at the desired output precision
 - Numerically stable
 - If precision lowered →
 - Faster
 - More energy efficient



AI-for-Science

Correctly leveraging reduced-precision data types can cut run-times and improve memory efficiency. Doing so safely requires knowing which variables tolerate reduced precision and which will silently corrupt your results. Offloading this effort to agents can greatly reduce the time a scientist spends developing code that is better spent doing their science.

Overview

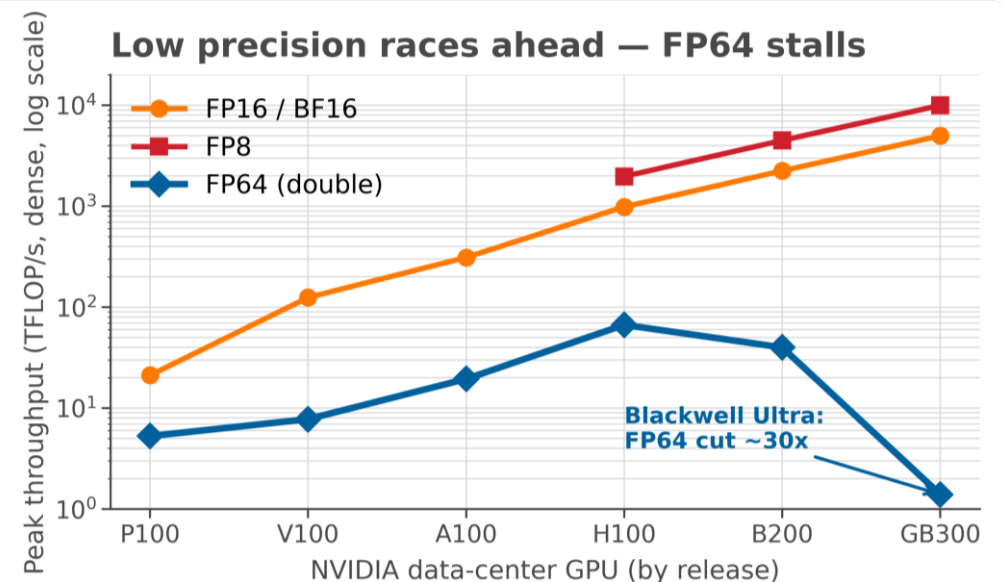
Motivation

Mixed-precision kernels are gaining importance. GPU vendors have poured their transistor budgets into low-precision throughput: FP16 and FP8 tensor performance has grown by orders of magnitude across recent generations, while peak FP64 performance has stalled and, on the newest parts, is actively shrinking. [1–4]. NVIDIA's Blackwell Ultra cuts hardware FP64 sharply relative to Hopper, and the gap between vendors is now a design choice, not a physical limit [3–5].

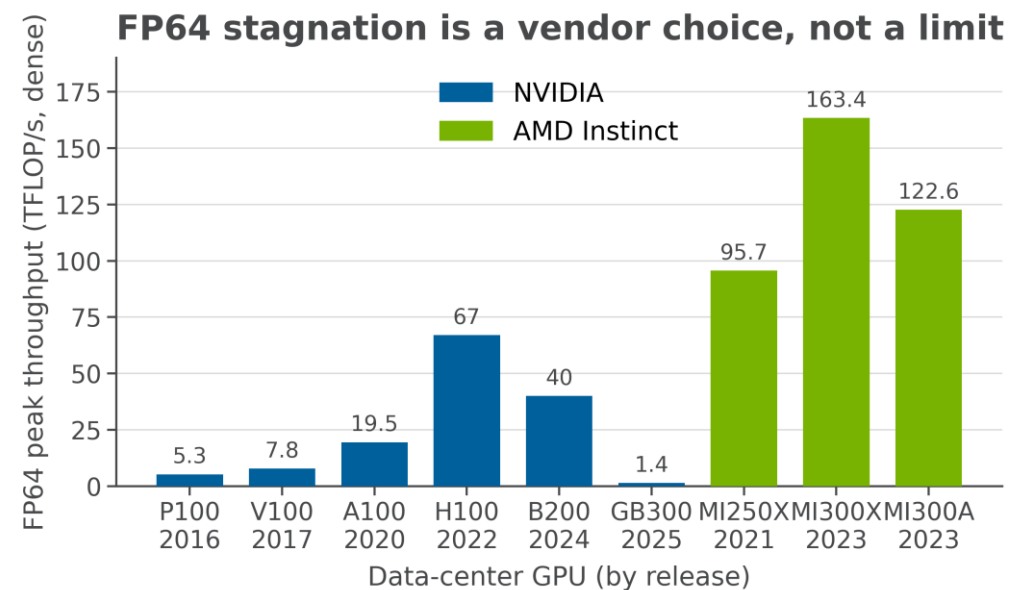


References

- [1] NVIDIA, “NVIDIA A100 Tensor Core GPU Architecture,” product datasheet, nvidia.com.
- [2] NVIDIA, “NVIDIA H100 Tensor Core GPU datasheet,” nvidia.com.
- [3] NVIDIA, “NVIDIA Blackwell / GB200 & GB300 NVL72 Datasheet,” nvidia.com.
- [4] NVIDIA, “NVIDIA B200 Tensor Core GPU,” product specifications, nvidia.com.
- [5] AMD, “AMD Instinct MI250X / MI300X / MI300A Accelerator Datasheets,” amd.com.



FP64/FP16 = Tensor-Core rate where available, else vector (P100, V100). GB300 FP64 derived from GB300 NVL72 rack spec (100 TFLOP/s / 72 GPUs). Dense, no sparsity.

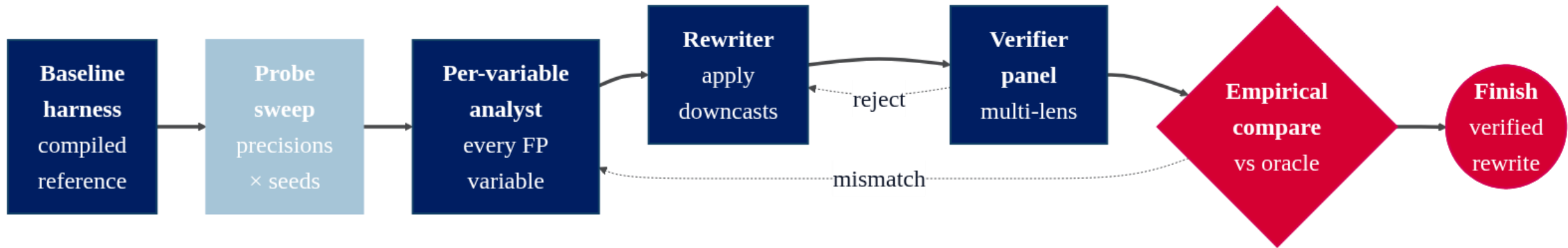


FP64 = Tensor/Matrix-Core rate where available, else vector. GB300 per-GPU derived from NVL72 rack spec. Dense, no sparsity.

Methods

Mermaid Diagram

UIC



An LLM orchestrator drives a fixed pipeline:

1. Generates a compiled reference harness
2. Empirically probes the kernel at three precisions, using the quad run as a high-precision oracle:
 - a) quad (FP128)
 - b) double (FP64)
 - c) single (FP32)
3. Reasons about every floating-point variable
4. Each proposed downcast set is tested against the oracle
 - a) Keep only if it meets the precision target
5. Each variable is tested first individually, then as a union, and bisected when interactions break the union
6. A verifier panel and a final numerical comparison gate the result

The workflow was tested for numerical stability and speed-up on four kernels with CPU and GPU.

- For this experiment, the tolerance was set to both sig_figs=6,10
- Kernels run at N=1M (horner_poly, series_exp) and N=16384 (nbody_potential, mixed_potential)
- Single CPU core (Kokkos::Serial)
- RTX 4060 GPU (CUDA sm_89).
- Timing is one warmup plus ten timed trials.
- Orchestrator and agents built on the Anthropic Claude SDK (claude-sonnet-4-6)

Results

Numerical Stability

Sig Figs = 6.

Across four FLOP-heavy kernels, the workflow's rewrites reproduce the FP128/FP64 reference to 6 significant figures in every case.



Definitions

Each kernel receives a verdict:

- **All-keep** means no variable could be safely dropped to FP32, so the rewrite stays entirely in FP64.
- **Mixed** means at least one variable was provably downcast to FP32, while the rest stay FP64.

Correctness — HOST (Kokkos::Serial)

verdict & agreement of each rewritten kernel vs. the quad (FP128) oracle

kernel	verdict (downcast)	agree @ sf6
horner_poly	all-keep	100000 / 100000
nbody_potential	MIXED eps2	4096 / 4096
series_exp	all-keep	100000 / 100000
mixed_potential	MIXED corr_acc, gauss	2048 / 2048

every rewrite reproduces the fp64/quad reference to 6 significant figures

Correctness — GPU (CUDA sm_89)

verdict & agreement of each rewritten kernel vs. the fp64 reference

kernel	verdict (downcast)	agree @ sf6
horner_poly	all-keep	1000000 / 1000000
nbody_potential	MIXED eps2, xi, yi, zi, dx, dy, dz, r2	16384 / 16384
series_exp	MIXED xi, term	1000000 / 1000000
mixed_potential	MIXED corr_acc, gauss	16384 / 16384

every rewrite reproduces the fp64/quad reference to 6 significant figures

Results

Numerical Stability

Sig Figs = 10.

Across four FLOP-heavy kernels, the workflow's rewrites reproduce the FP128/FP64 reference to 10 significant figures in every case.



Definitions

Each kernel receives a verdict:

- **All-keep** means no variable could be safely dropped to FP32, so the rewrite stays entirely in FP64.
- **Mixed** means at least one variable was provably downcast to FP32, while the rest stay FP64.

Correctness — HOST (Kokkos::Serial)

verdict & agreement of each rewritten kernel vs. the quad (FP128) oracle

kernel	verdict	agree @ sf10
horner_poly	all-keep	100k / 100k
nbody_potential	all-keep	4096 / 4096
series_exp	all-keep	100k / 100k
mixed_potential	MIXED	2048 / 2048

every rewrite reproduces the fp64/quad reference to 10 significant figures

Correctness — GPU (CUDA sm_89)

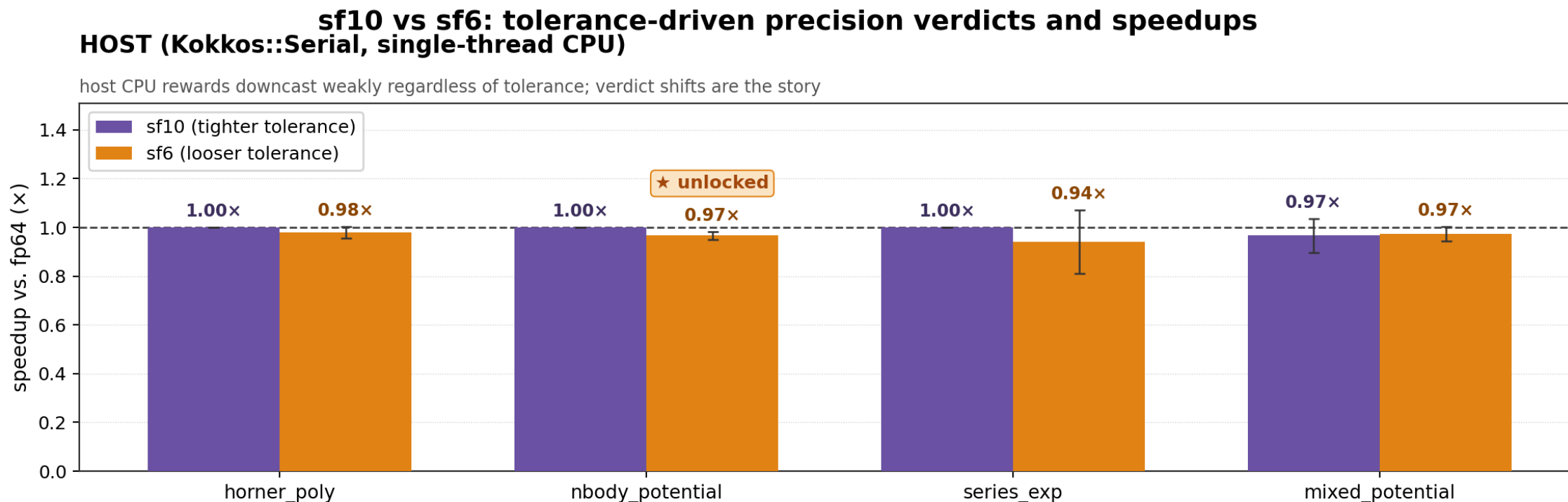
verdict & agreement of each rewritten kernel vs. the fp64 reference

kernel	verdict	agree @ sf10
horner_poly	all-keep	1M / 1M
nbody_potential	all-keep +Kahan	16384 / 16384
series_exp	all-keep	1M / 1M
mixed_potential	MIXED	16384 / 16384

every rewrite reproduces the fp64/quad reference to 10 significant figures

There is no speed-up on the CPU from lowering precision. This is because either:

- (a) CPUs don't have hardware-accelerated execution units for low-precision math
- (b) These kernels are bottlenecked by memory transfer, i.e., memory-bound

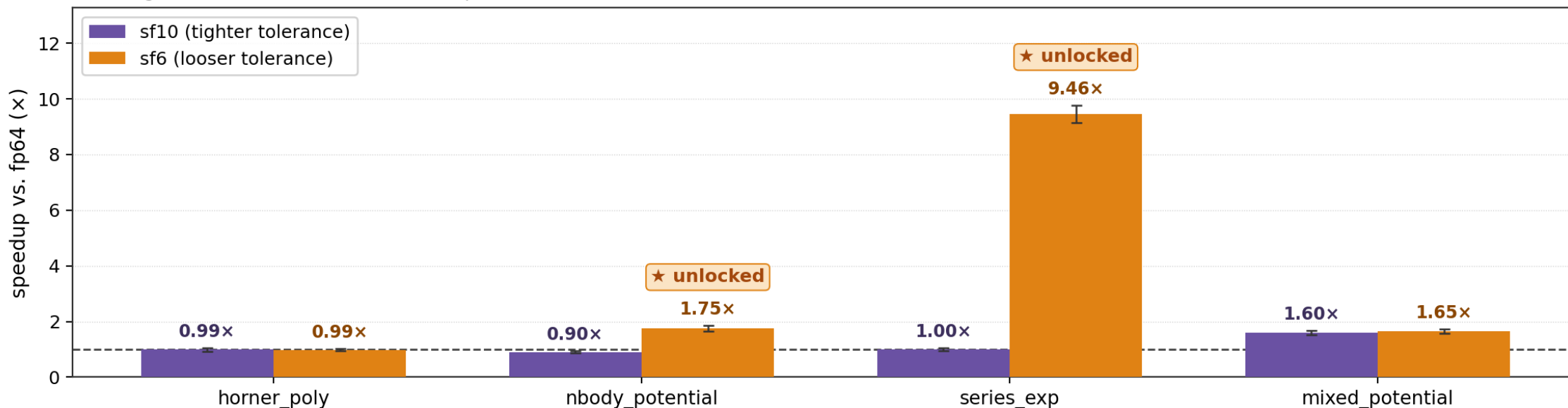


On the GPU, mixed-precision kernels see a significant speed-up compared to kernels that remain in all-double precision. This is especially notable in cases where sf=6 obtained a mixed verdict, but sf=10 obtained an all-keep verdict (nbody_potential, series_exp).

The mixed-precision kernels can take advantage of hardware-accelerated execution units, resulting in speedups.

sf10 vs sf6: tolerance-driven precision verdicts and speedups
GPU (CUDA sm_89, RTX 4060)

loosening tolerance from sf10 to sf6 unlocks fp32 downcasts in 2 of 4 GPU kernels



The workflow maintains numerical stability and can generate speed-up in cases with safe downcasts.

- Automated agentic tuning works: across all four kernels, the workflow's rewrites matched the FP128/FP64 oracle to 6,10 significant figures.
- Empirical per-variable gating is decisive: only provably safe downcasts are kept, and kernels with no safe downcast are correctly left in FP64.
- Speedups are real but hardware-bound: the genuinely mixed kernel ran $1.60 \pm 0.09\times$ faster on GPU (FP64+FP32), while software quad on CPU is 10–244 $\times$ slower. GPUs are where mixed precision pays off.

Create a more robust workflow that runs at scale. This can lead to greater time and energy savings.

- Add a kernel-extractor stage to pull individual kernels out of full application source and tune them automatically.
- Broaden the emulation library (float-float / Dekker) for kernels that need more than a plain downcast.
- Migrate the compile and run stages to remote schedulers (JLSE) for at-scale evaluation. At scale, speed-ups from using mixed-precision kernels can compound, likely saving time and energy.



Thank you for listening