A Distributed Mixed-Precision N-Body Engine: Scaling Behavior and the Precision Limits of a Direct-Summation Kernel

Tenicka Norwood, Michael Flanders, Naw Safrin Sattar
Computer Science and Data Science Department, School of Applied Computational Sciences
Meharry Medical College, Nashville, Tennessee, USA
tenicka.norwood@mmc.edu, michael.flanders@mmc.edu, nawsafrin.sattar@mmc.edu

***Abstract*—We report a distributed direct-summation N-body engine used to study two coupled limits: where the single collective per step begins to limit strong scaling, and where reduced-precision arithmetic begins to corrupt the physics. At a fixed 32768 bodies on one 96-core node, the kernel scales almost ideally to 16 ranks and reaches a 57-fold speedup at 96. In-core phase timers show that the loss beyond 16 ranks comes from one straggling rank, whose peers wait at the collective, rather than from data transfer, which stays under 1% of a step. A precision ladder from FP64 down to FP8 shows that the accumulator, not the individual force terms, sets the error floor.**

***Index Terms*—N-body simulation, mixed precision, MPI, strong scaling, floating-point summation**

I. Introduction

Direct summation is the accuracy reference for gravitational N-body simulation [1]: every pair interaction is computed, without the approximation that tree codes introduce [2]. Its $O(N^2)$ arithmetic on $O(N)$ data makes the kernel compute-bound and a natural target for reduced precision [3], including FP8 [4]. GPU N-body codes have long computed forces in single precision [5] or in emulated double-single arithmetic to recover accuracy [6], and compensated summation [7], [8] repairs a low-precision accumulator in software at extra cost per term. Molecular dynamics codes go further: SPFP computes forces in single precision but accumulates them in 64-bit fixed point [9].

We measure that split directly for gravitational N-body, rounding each pairwise term and the running sum independently, and ask where the single collective per step stops being free under strong scaling.

II. Design and Methods

Engine. The engine is a single C++17 binary using MPI and OpenMP. Each of p ranks owns a contiguous block of N/p bodies. A step is a kick-drift-kick leapfrog update, which is symplectic, so energy error stays bounded and any secular drift reflects the arithmetic rather than the integrator [10]. Positions are packed into one buffer and exchanged with a single `MPI_Allgatherv`; each rank then evaluates its $O(N^2/p)$ block of softened forces. Initial conditions are a Plummer sphere [11] whose energy matches the analytic value to 0.5% at $N = 16384$.

Measurement. Compute and communication are timed separately inside the step, in barriered windows, and reduced across ranks with `MPI_MAX`.

Precision model. FP64 and FP32 run natively. TF32, FP16, BF16 and two FP8 formats are emulated by rounding every operation in a pairwise term to the target format [12]; their timings are discarded. Each emulated format runs two arms: rounded terms with an FP64 accumulator, and rounded terms with the accumulator rounded too. The second admits stagnation [8]: once the partial sum exceeds a term by more than $1/u$, the term is lost. Force error is measured at $t = 0$ against an FP64 oracle, because the dynamics is chaotic and late-time trajectory differences measure Lyapunov growth rather than rounding.

Setup. Runs used ACES: 96-core Intel Xeon Platinum 8468 nodes, Intel MPI 2021.9, and `-fp-model=precise` so the FP64 oracle is not reassociated.

TABLE I
Relative force error at $N = 4096$ (mean of 3 seeds)

Format	u	FP64 accum.	Target accum.	Penalty
FP32	6.0×10^{-8}	1.5×10^{-7}	–	–
TF32	4.9×10^{-4}	4.6×10^{-4}	5.1×10^{-2}	111
FP16	4.9×10^{-4}	5.5×10^{-4}	5.1×10^{-2}	92
BF16	3.9×10^{-3}	3.6×10^{-3}	4.4×10^{-1}	125
FP8-E5M2	1.3×10^{-1}	2.1×10^{-1}	8.7×10^{-1}	4
FP8-E4M3	6.3×10^{-2}	1.0	1.0	1

III. Results

Precision. Table I gives the ladder at $N = 4096$ and Fig. 1 plots it. With an FP64 accumulator, force error tracks the unit roundoff of each format. Rounding the accumulator as well costs a further 92 to 125 times at TF32, FP16 and BF16, and the penalty grows with N: at $N = 8192$ it reaches 214 to 458 times. FP8-E4M3 fails in both arms.

Scaling. At a fixed $N = 32768$ on one node, FP64 efficiency is 99% through 8 ranks and 98% at 16, and the speedup

This work used ACES advanced computing resources at Texas A&M High Performance Research Computing through allocation CIS260987 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

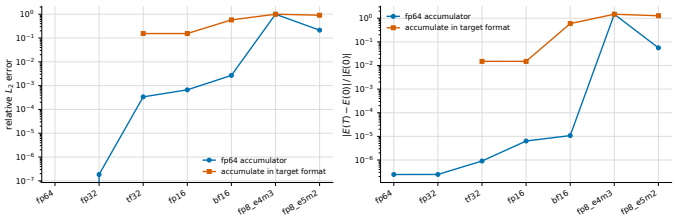


Fig. 1. Precision ladder. Left: force error at $t = 0$. Right: energy drift.

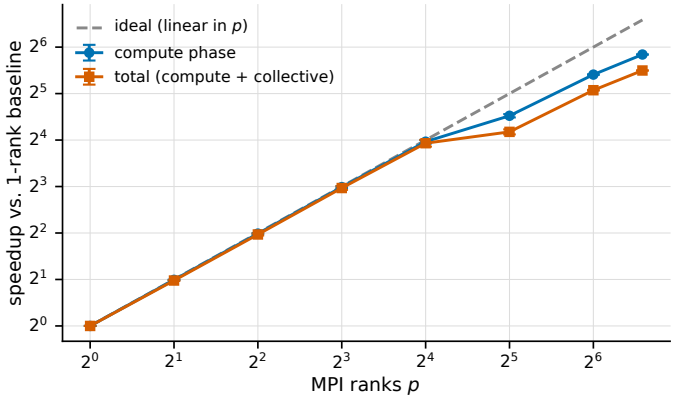


Fig. 2. Strong scaling, $N = 32768$. Left: efficiency (dashed: as first reported). Right: slowest rank relative to ideal.

at 96 ranks is 57 (Fig. 2). Beyond 16 ranks, the slowest rank computes 1.4 times longer than its peers with identical work, and the time the other ranks spend waiting for it at the collective first appeared as a 21% communication share. A follow-up run that times arrival skew separately from transfer measured transfer below 1% of a step, and step time within 1% of the slowest rank's compute time, which is the step time we report. FP32 runs showed no straggler and held 80% efficiency at 96 ranks against 60% for FP64, running 2.9 times faster.

IV. Conclusions

Precision is limited by the reduction, not the operands: keep the accumulator in FP64 and spend low precision on the terms. On one node the collective is nearly free, and per-rank timers located the scaling loss in a straggling rank that a whole-loop timer would have attributed to communication. Next steps are identifying the straggling core, a 1024-rank run across 11 nodes, and pairwise and stochastic-rounding accumulators.

References

[1] S. J. Aarseth, *Gravitational N-Body Simulations: Tools and Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[2] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, pp. 446–449, 1986.

[3] N. J. Higham and T. Mary, "Mixed precision algorithms in numerical linear algebra," *Acta Numerica*, vol. 31, pp. 347–414, 2022.

[4] P. Micikevicius *et al.*, "FP8 formats for deep learning," 2022, arXiv:2209.05433.

[5] L. Nyland, M. Harris, and J. Prins, "Fast N-body simulation with CUDA," in *GPU Gems 3*, H. Nguyen, Ed. Boston, MA, USA: Addison-Wesley, 2007, ch. 31, pp. 677–695.

[6] E. Gaburov, S. Harfst, and S. Portegies Zwart, "SAPPORO: A way to turn your graphics cards into a GRAPE-6," *New Astronomy*, vol. 14, no. 7, pp. 630–637, 2009.

[7] W. Kahan, "Pracniques: Further remarks on reducing truncation errors," *Commun. ACM*, vol. 8, no. 1, p. 40, 1965.

[8] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: SIAM, 2002.

[9] S. Le Grand, A. W. Götz, and R. C. Walker, "SPFP: Speed without compromise—A mixed precision model for GPU accelerated molecular dynamics simulations," *Comput. Phys. Commun.*, vol. 184, no. 2, pp. 374–380, 2013.

[10] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the Störmer–Verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

[11] H. C. Plummer, "On the problem of distribution in globular star clusters," *Mon. Not. R. Astron. Soc.*, vol. 71, pp. 460–470, 1911.

[12] N. J. Higham and S. Pranesh, "Simulating low precision floating-point arithmetic," *SIAM J. Sci. Comput.*, vol. 41, no. 5, pp. C585–C602, 2019.