Optimize Memory Usage in Vector Particle-In-Cell (VPIC) to Break the 10 Trillion Particle Barrier in Plasma Simulations

Nigel Tan[1], Robert Bird[2], Guangye Chen[2], and Michela Taufer[1]

[1] University of Tennessee, Knoxville TN 37919, USA
[2] Los Alamos National Laboratory. Los Alamos, NM, 87545, USA

Abstract. Vector Particle-In-Cell (VPIC) is one of the fastest plasma simulation codes in the world, with particle numbers ranging from one trillion on the first petascale system, Roadrunner, to ten trillion particles on the more recent Blue Waters supercomputer. As supercomputers continue to grow rapidly in size, so too does the gap between compute capability and memory capability. Current memory systems limit VPIC simulations greatly as the maximum number of particles that can be simulated directly depends on the available memory. In this study, we present a suite of VPIC memory optimizations (i.e., particle weight, half-precision, and fixed-point optimizations) that enable a significant increase in the number of particles in VPIC simulations. We assess the optimizations' impact on a GPU-accelerated Power9 system. Our optimizations enable a 31.25% reduction in memory usage and up to 40% increase in the number of particles.

Keywords: Particle-In-Cell method · Mixed-precision · Fixed-point · Plasma physics.

1 Introduction

Vector Particle-In-Cell (VPIC) is a high-performance particle-in-cell code that models plasma phenomena such as magnetic reconnection, fusion, solar weather, and laser-plasma interaction [2]. VPIC is one of the fastest PIC codes in the world and has performed some of the largest plasma simulations in history, ranging from one trillion particles on the first petascale system, Roadrunner [3], to ten trillion particle simulations on the more recent Blue Waters supercomputer [5]. VPIC simulations use large numbers of particles (i.e., on the order of trillions of particles [3]) to model real world phenomena. As we move the VPIC code from CPUs to accelerators (i.e., GPUs), the number of particles in VPIC simulations become limited by memory rather than compute capabilities; modern CPUs can access up to 4 TB of memory while GPUs (such as Nvidia A100) are limited to 80 GB, a factor of 50 difference. Moreover, data movement between CPUs and GPUs is costly, with PCIe 4.0 limited to 32GB/s in one direction. Specialized protocols and hardware have been developed to help address

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_35

the issue; NVLink 3.0 achieves up to 300 GB/s in one direction [10]. Hardware and software techniques for maximizing communication efficiency are a major ongoing field of research [18]. At the code level, running VPIC on modern supercomputers with accelerators, while scaling up the number of particles, requires rethinking how the code uses memory.

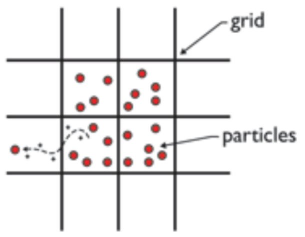
In this paper, we introduce a new particle representation and develop a suite of optimizations for VPIC's particle storage format (i.e., particle weight, half-precision position, and fixed-point position optimizations) to tackle the memory usage problem. Our new particle representation reduces particle memory usage by up to 31.25%. We demonstrate that our optimizations enable significantly larger simulations, and that the optimized simulations produce accurate scientific results. Section 2 introduces VPIC's particle representation and workflow; Section 3 describes our method to increase particle count by reducing memory usage; Section 4 presents our performance and accuracy tests; Section 5 provides an overview of existing plasma simulation codes; and Section 6 summarizes the key results and introduces future work.

2 VPIC Particles and Workflow

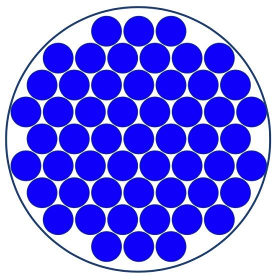
VPIC is a high-performance implementation of the particle-in-cell method used for plasma simulations [11]. VPIC operates by defining a simulation space divided into a grid of cells and modeling particle movement across the cells. In other words, particles are distributed across an n-dimensional (n-D) space that is decomposed into an n-D grid. The resolution of the grid determines cell size and the maximum time step length. Figure 1a shows an example of a 2-D grid. Each simulated particle is a macroparticle (Figure 1b) with a defined weight (i.e., the number of real particles modeled by each macroparticle).

The grid resolution and the number of particles both affect simulation accuracy and computational costs. Fine resolution grids better approximate a continuous n-D space. However, such fine resolution grids increase computational costs due to field operations, and the time step size must shrink accordingly to ensure numerical stability. Thus the number of time steps increases for the same period of simulated time, which further increases computational costs. Increasing the number of particles can also improve accuracy by more closely modeling real world plasma phenomena. High particle count primarily affects the particle advance stage with computational costs scaling linearly with the number of particles. In standard PIC simulations, the cell size is close to the Debye length (which is the smallest physical length scale in a plasma), and the time step is set as large as possible. Between the number of time steps and particle count, increasing particle count is generally preferred, as increasing the number of time steps incurs higher computational and communication costs.

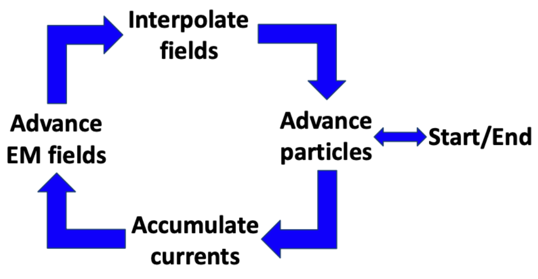
A VPIC simulation is an iterative process across a defined number of time steps: each iteration has four key stages (as shown in Figure 1c). First, the electric and magnetic fields are gathered from the grid points to each particle's location (interpolate fields). Second, particles move around based on the forces calculated from the electric and magnetic fields (advance particles). Third, the current generated by the particles' movements is scattered for each cell (accumulate currents). Last, electric and magnetic fields are advanced based on the accumulated current (advance EM fields), and the next iteration starts. For large, particle-heavy simulations, the first three stages take most of the simulation time, as each of these stages must operate on all the particles. In the last stage, VPIC advances the electric and magnetic fields at each grid point. The field advance stage is cheaper compared to prior stages since the number of particles tends to greatly outnumber the number of grid points. Furthermore, past I/O studies using VPIC noted that particles are responsible for the vast majority of memory usage. The trillion particle run in [6] required only 80 GB of storage for the electric and magnetic fields compared to the approximately 30 TB necessary for the particles. The 375-fold difference in memory requirement demonstrates that particle storage is a bottleneck for large scale VPIC simulations.



(a) Grid and particles



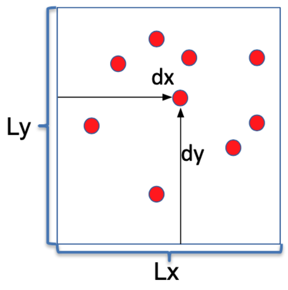
(b) Macroparticle



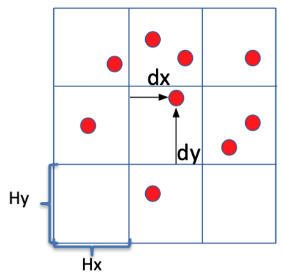
(c) VPIC iterative algorithm

Fig. 1: VPIC features: grid and particles (a); concept of macroparticle (b); and VPIC stages (c).

The particle position can be stored in two different ways based on the coordinate system: globally and locally. Global coordinates of particles are calculated and stored based on their absolute position in the n-D space. In Figure 2a the global positions of the particles are depicted in a 2-D space as (dx, dy). Alternatively, particle positions can be stored in local coordinates with each particle storing its cell index and position within the cell. In Figure 2b), each particle position in the 2-D space is defined as $(i*H_x+dx, j*H_y+dy)$, where (i,j) is the cell index, (H_x, H_y) is the cell size, and (dx, dy) is the local position of the particle. This representation requires extra space for the cell index but allows VPIC to represent each particle position more accurately. This is because the distribution of floating-point values follows an almost logarithmic distribution [16]. Values further away from zero are less precise, and approximately half of all floating-point values exist in $[-1, 1]$. Thus, local coordinates within the smaller cell have a more even and dense distribution of values to represent a particle position. The original VPIC code uses local coordinates and single-precision for storing particle positions.



(a) (dx, dy)



(b) $(i*Hx+dx, j*Hy+dy)$

Fig. 2: Global and local particle coordinates.

3 Increase Particle Count by Reducing Memory Usage

In VPIC simulations, the larger the number of particles, the more physically accurate the simulations and the greater the memory usage. Each particle is a macroparticle of 32 bytes (as shown in Figure 3) that comprises 3 floats for position, 3 floats for momentum, 1 integer for the cell index, and 1 float for weight. There is a major disparity in memory usage between particle data and all the other data used in the code. Figure 4 shows an example of disparity for a VPIC simulation of 512 particles per cell; the particles are responsible for over 90% of the total memory usage. Larger VPIC simulations have thousands (or more) of particles per grid cell [3], making particles the primary focus when it is time to optimize memory usage. For instance, the simulations described in [1], which study laser-plasma modeling, have up to $65,536$ particles per grid cell.

```
struct particle {
  float w;           // Weight
  float dx, dy, dz; // Position
  int i;             // Cell index
  float ux, uy, uz; // Momentum
};
```

Fig. 3: Original VPIC particle data structure.

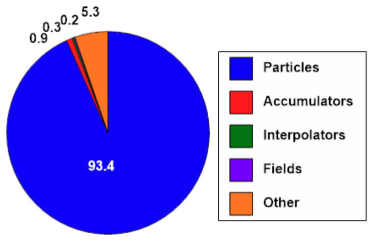


Fig. 4: Breakdown of memory usage for a simulation using 512 particles per cell.

3.1 Types of Optimizations

Our suite of optimizations reduce memory usage associated with particles' weight and position through reduced precision (i.e., half-precision) and alternative number representation formats (i.e., fixed-point). Accuracy is maintained by leveraging simulation properties and characteristics of the particle-in-cell method.

Table 1 describes the various types of precision formats for floating-point numbers supported by existing architectures. The original VPIC code uses single-precision by default (i.e., float), as shown in Figure 3. The three candidates for reduced storage formats are FP16, TensorFloat [20], and Bfloat16 [17]. Compared to FP16, TensorFloat requires more storage (19 bits), and Bfloat16 loses precision (decimal digits). Thus we use FP16 for our optimizations.

Precision	Sign	Exponent	Fraction	Decimal Digits
Double (FP64)	1	11	52	≈ 15.9
Single (FP32)	1	8	23	≈ 7.2
Half (FP16)	1	5	10	≈ 3.3
TensorFloat	1	8	10	≈ 3.3
Bfloat16	1	8	7	≈ 2.4

Table 1: Floating-point formats with their numerical representations.

3.2 Particle Weight

In VPIC, the macroparticle's weight represents the number of real particles modeled by the simulated particle. The particle weight either changes within a limited range or does not change at all during a simulation. We use this property to optimize memory usage by adjusting how weight is stored.

When the particle weight varies but has a limited range of values, we can replace the single-precision weight with a 16-bit integer denoting a multiple of a known base weight. Alternatively, the 16-bit integer can be an index for a lookup table of particle weights. Both methods allow 65,536 different weights while reducing particle memory by 6.25%. We call this optimization short weight (SW).

When the particle weight remains constant for all particles of the same species throughout a simulation, particle weight can be removed entirely and replaced with a per species constant weight. This solution reduces memory usage by 12.5% over original VPIC. We call this optimization constant weight (CW). Figure 5 shows the reduction in memory usage for both optimizations (i.e., SW and CW).

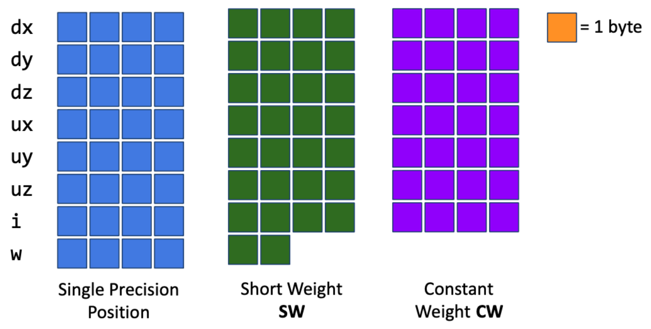


Fig. 5: Particle memory usage for default single-precision VPIC and VPIC with our short weight (SW) and constant weight (CW) optimizations.

3.3 Particle Position

The position of a particle in the original VPIC code is represented by three float values in single-precision (32 bits). We optimize the code by switching the representation to half-precision (16 bits). Figure 6 shows that by deploying half-precision we can reduce memory usage by 18.75% compared to the original VPIC and 31.25% when combined with the constant weight optimization (or CW). For a fine resolution grid, half-precision can produce simulations of sufficient accuracy while enabling larger simulations, as we will show in Section 4.

3.4 From Half-Precision to Fixed-Point

Half-precision particle position may incur an accuracy penalty in comparison to the default single-precision. Particle positions in each cell in VPIC using local particle coordinates are normalized to [-1,1], and thus we can use 16-bit fixed-point numbers instead of half-precision to minimize the loss in accuracy [7]. Fixed-point numbers allow us to maximize the number of bits used for precision. The fixed-point $Qm.n$ format specifies m bits for the integer and n bits for the fractional portion. We use the $Q1.14$ fixed-point format for storing position. One

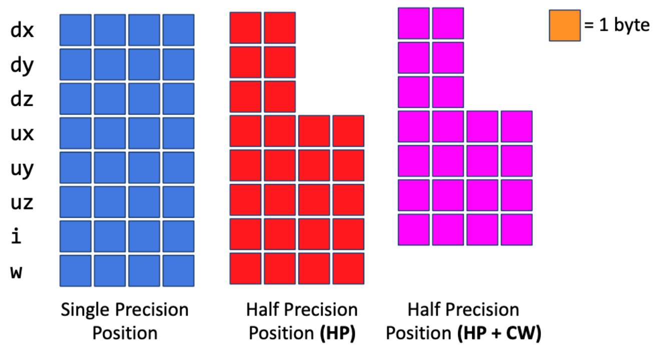

dx
dy
dz
ux
uy
uz
i
w
= 1 byte
Single Precision Position
Half Precision Position (HP)
Half Precision Position (HP + CW)

Fig. 6: Particle memory usage comparison between original VPIC, VPIC with our half-precision (HP) optimization, and VPIC with both half-precision and constant weight optimizations applied (HP+CW).

bit is for the sign, one bit for the integer portion, and the remaining 14 bits for the fractional portion. The $Q1.14$ format uses the same amount of memory as half-precision but has an additional four bits (approximately one decimal digit) for improving accuracy.

3.5 Particle Momentum and Cell Index

Particle momentum and cell index are represented by three single-precision floats and a 32-bit integer respectively. Momentum is left unchanged in this work. Momentum values are normalized to c (speed of light). Initial tests indicate that switching the momentum values from single-precision to half-precision would result in insufficient accuracy. Particle cell index determines which cell the particle resides in and is kept at the default 32-bit integer. A short 16-bit integer has insufficient range for large-scale simulations.

4 Performance and Accuracy

We present four test scenarios. The first test models laser-plasma interaction and is used for measuring runtime performance and memory usage of the original VPIC and the optimized version. The remaining three tests constitute a set of simple benchmark problems for analyzing the impact of our optimizations on the VPIC's numerical accuracy. All the tests were conducted on a four-node, GPU-accelerated IBM Power9 system. Each node has 155 GB of memory and 32 cores with 128 threads; two nodes host two Nvidia Tesla V100 GPUs each for a total of 4 GPUs. A single GPU is used, which is sufficient for testing and simplifies both data collection and analysis.

4.1 Performance

For testing runtime and memory performance, we have four problem sizes designed to use the GPU's full 16 GB memory capacity. Problem size is determined by the total number of particles in the system. The base case requires 16 GB to run using the original single-precision VPIC. The remaining cases increase in size until only our optimized versions can successfully run. Runtime tests are repeated 10 times and the average runtime is used to compare configurations. Memory usage is measured using the Space Time Stack tool provided by the Kokkos Tools profiling utilities [12]. The tool tracks Kokkos allocations and the maximum memory usage on the GPU.

We test VPIC scalability in terms of the number of particles using simulations modeling laser-plasma interactions. Figure 7 shows the scalability for the original VPIC code and when using our four optimizations (i.e., with short particle weight (SW), with constant particle weight (CW), using half-precision particle position (HP), and using fixed-point particle position (FP)). Missing columns indicate failed simulations that crash due to insufficient memory. We control problem size by adjusting the number of particles per cell (Nppc). The grid resolution and number of time steps are kept constant.

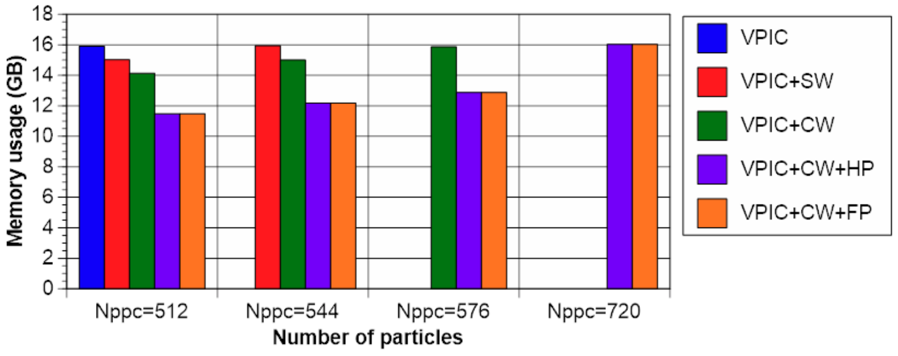


Fig. 7: Memory usage in GB of original VPIC and VPIC with our optimizations. Number of particles is proportional to the number of particles per cell (Nppc).

Our optimizations demonstrate a significant reduction in memory usage, as shown in Figure 7. The optimizations to particle position (HP and FP) provide the greatest reduction in memory usage; when combined with constant weight (CW), they can increase the total number of particles by a factor of up to 40%. Figure 8 demonstrates that our optimizations also have minimal effect on runtime performance. Specifically, no negative effect is observed on runtime performance; in fact runtime can improve thanks to the hardware needing to move less data through its memory system and between processors. In other worlds, the optimized VPIC minimizes the amount of data movement between CPU and GPU. This results in a relatively small improvement in performance

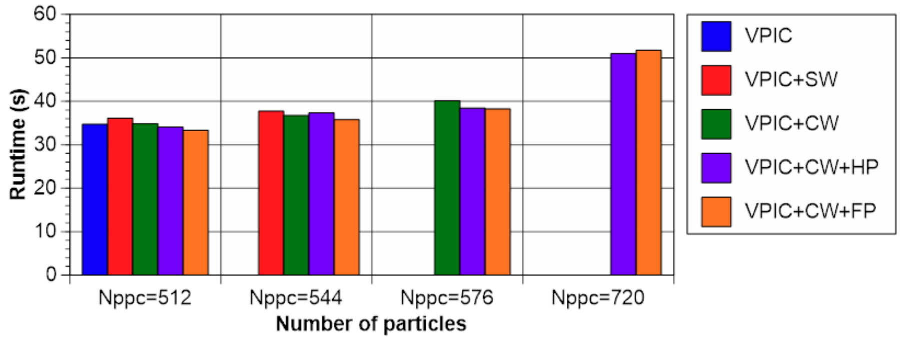


Fig. 8: Runtime in seconds of original VPIC and VPIC with our optimizations.

that largely cancels out the additional overhead from converting data between different formats when entering and exiting the advance particle stage (i.e., from single-precision to half-precision and vice versa, from floating-point to fixed-point and vice versa) where the optimizations are deployed.

4.2 Accuracy

Measuring VPIC accuracy is difficult due to the lack of rigorous error quantification for the PIC method. Numerical accuracy testing is conducted with three different benchmark problems. In each test, we compare the original single-precision VPIC against VPIC with our half-precision and 16-bit fixed-point optimizations.

The first benchmark problem is a simple 1-D problem with periodic boundary conditions and an arbitrary number of particles N_p. Each particle has a constant weight $\frac{1}{N_p}$ and an initial momentum of $V_0 = 0$. The initial potential and electric field are given by

$$\phi_i(x) = \begin{cases} x(1-x_i),\, 0 < x < x_i \\ x_i(1-x),\, x_i < x < 1 \end{cases}$$

$$\phi_b(x) = \frac{x^2 - x}{2}, \quad \phi(x) = \sum_{i=1}^{N_p} \phi_i(x) w_i + \phi_b(x), \quad E(x) = -\frac{\partial \phi}{\partial x}$$

This test has an analytical solution for particle position described by

$$x_i = \frac{1}{N_p}\left[\left(\sum_k x_{k0}\right) - \frac{N_p+1}{2} + i\right] - \frac{1}{N_p}\left[\sum_k (x_{k0} - x_{i0}) - \frac{N_p+1}{2} + i\right]\cos(t)$$

Using the analytical solution we can track errors in an individual particle's position over time. For simplicity, we set the number of particles to two.

Table 2 describes the relative memory usage, and accuracy for each particle position representation. The 16-bit fixed-point setting guarantees an additional decimal digit of precision compared to 16-bit floating point.

Format	Space reduction (# times)	Accuracy (Digits)
32-bit floating-point	1.00x	7.225
16-bit floating-point	0.81x	3.311
16-bit fixed-point	0.81x	4.515

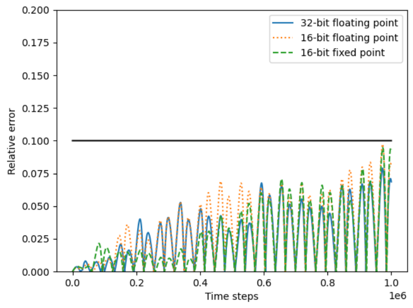
Table 2: Relative memory usage, run-times, and accuracy for each potential particle position representation.

This two-particle test problem can demonstrate the effects of our optimizations combined with grid resolution on particle position accuracy. Relative position error is tracked across 1,000,000 time steps with two different grid resolutions (i.e., $10,000$ and $6,000$ cells). The accuracy results for the two resolutions are shown in Figure 9a and Figure 9b respectively. Both the 16-bit floating-point and the 16-bit fixed-point formats trade storage space for precision as shown theoretically in Table 2: 16-bit floating-point maintains 3.311 decimal digits of precision while 16-bit fixed-point maintains 4.515 digits, a significant drop from the 7.225 digits for 32-bit floating-point. Figure 9a demonstrates that a sufficiently high resolution grid can make up for the drop in accuracy when using 16-bit floating-point or fixed-point for particle position.

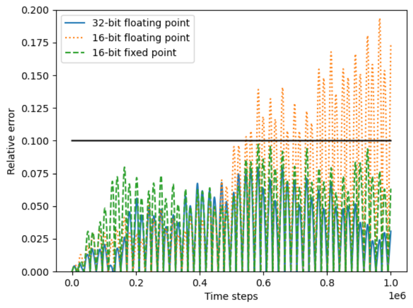
It is important to note that grid resolution, time step length, and simulation length are directly related. High resolution grids cause the time step length to shorten, which in turn causes the number of time steps and the overall runtime to increase for a fixed simulation length. As a result, most simulations keep grid resolution as coarse as possible to minimize the number of simulated time steps. Striking a balance between grid resolution and accuracy is important. Figure 9b shows the same test with a lower resolution grid (i.e., 6,000). The effect of grid resolution on accuracy is clearly shown by the 16-bit floating-point format failing to maintain sub 10% relative error in this test. The fixed-point format has a notable increase in accuracy over half-precision and is comparable to the 32-bit single-precision floating-point; both 16-bit fixed-point and 32-bit floating-point achieve sub 10% relative error throughout the test Our results indicate that 16-bit fixed-point not only enables larger simulations than 32-bit single-precision but also improves accuracy over half-precision.

The second benchmark problem models the two-stream instability, which is an electrostatic instability commonly seen in plasma physics. Two counter streaming electron beams move in a stationary ion background [22]. The streams are vulnerable to electrostatic perturbations that result in charge bunching behavior [21]. The simulation is based on the two-stream instability deck described in [9]. The two-stream test focuses on verifying the conservation of momentum. In Figure 10a, we observe how the conservation of momentum is maintained extremely well by all three variations of VPIC for the first 600 time steps. After 600 time steps, errors for fixed-point start growing while the errors for 32-bit and 16-bit floating-point remain low for a few dozen time steps before growing and eventually plateauing. Half-precision and 16-bit fixed-point notably plateau with lower relative error than single-precision.

The third benchmark problem simulates the Weibel instability [15]. Similar to the two-stream instability, the Weibel instability can arise from counter



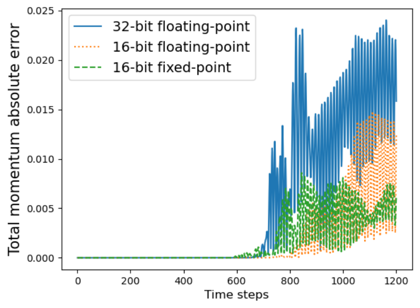
(a) 10,000 cells



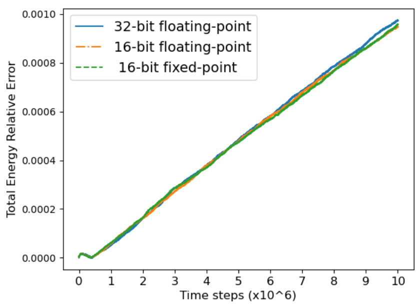
(b) 6,000 cells

Fig. 9: Accuracy comparison VPIC simulations with different grid resolution. Particle weight is kept constant and particle position is shown using 32-bit floating-point, 16-bit floating-point, and 16-bit fixed-point.

streaming beams. Unlike the two-stream instability, the Weibel instability has electromagnetic perturbations that result in elongated structures in the plasma (filamentation). The process converts the kinetic energy of the particle beams into magnetic field energy [19]. The purpose of this test is to verify that energy is conserved with the particle format optimizations applied. Figure 10b clearly shows that total energy is conserved very well with less than 0.1% relative error across 10 million time steps for the Weibel instability. The curves for single-precision, half-precision, and fixed-point are tightly grouped together. Thus with our optimizations applied, conservation of energy is upheld with minimal variation compared to original VPIC.



(a) Two-stream instability.



(b) Weibel instability.

Fig. 10: Conservation of momentum relative error for the two-stream instability (a) and conservation of energy relative error for the Weibel instability.

5 Related Work

Current state-of-the-art plasma simulation codes include VPIC, OSIRIS [13], WarpX [23], and PIConGPU [4]. These codes have all been used to run plasma simulations with approximately 10 trillion particles at the largest scale [5], [14], [8]. Both WarpX and PIConGPU use a similar particle representation format as VPIC. All three codes use single-precision for performance reasons. To the best of our knowledge, no other particle-in-cell code has investigated the use of half-precision or fixed-point representation for optimizing particle format.

6 Conclusions and Future Work

This paper demonstrates how the combination of constant weight and lower precision position in the VPIC code reduces memory usage by up to 31.25% and enables up to 40% increase in the number of particles using the same amount of memory for particle-heavy simulations. The optimizations improve performance by reducing the amount of data movement, which compensates for any additional operations introduced. Unlike other work in mixed-precision algorithms, our optimizations use reduced precision for storage rather than accelerating computation. We also show that our optimizations not only greatly increase simulation scale on memory constrained hardware, but can also achieve similar accuracy as the original single-precision VPIC. The fixed-point optimizations, in particular, show great promise thanks to the higher precision compared to half-precision. The higher precision allows for lower grid resolutions which ultimately decreases the number of time steps in the simulation. It is important to note that our optimizations may require changes to simulation parameters, namely the grid resolution. A grid resolution that produces accurate results for single-precision VPIC may not be accurate enough with half-precision or fixed-point. Such simulations may need to be adjusted to fully benefit from our optimizations.

Future work includes studying the algorithmic changes necessary to enable lower precision storage for particle momentum and the impact of these changes on both scalability and accuracy. We also plan to develop heuristics and methodologies to help physicists use our suite of optimizations. Adjustments to simulation parameters (e.g., grid resolution, step size and number of steps) are necessary to maintain accuracy. Different classes of plasma simulations also need to be studied to better understand which classes benefit from our optimizations the most and what changes to simulation settings are required to use our optimizations.

Acknowledgments

Work performed under the auspices of the U.S. DOE by Triad National Security, LLC, and Los Alamos National Laboratory(LANL). This work was supported the LANL ASC and Experimental Sciences programs. The UTK authors acknowledge the support of LANL under contract #578735 and IBM through a Shared University Research Award. LA-UR-21-21297

References

1. Arber, T., Bennett, K., Brady, C., Lawrence-Douglas, A., Ramsay, M., Sircombe, N., Gillies, P., Evans, R., Schmitz, H., Bell, A., et al.: Contemporary particle-in-cell approach to laser-plasma modelling. Plasma Physics and Controlled Fusion **57**(11), 113001 (2015)
2. Bowers, K.J., Albright, B., Yin, L., Bergen, B., Kwan, T.: Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation. Physics of Plasmas **15**(5), 055703 (2008)
3. Bowers, K.J., Albright, B.J., Bergen, B., Yin, L., Barker, K.J., Kerbyson, D.J.: 0.374 pflop/s trillion-particle kinetic modeling of laser plasma interaction on road-runner. In: SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. pp. 1–11. IEEE (2008)
4. Burau, H., Widera, R., Hönig, W., Juckeland, G., Debus, A., Kluge, T., Schramm, U., Cowan, T.E., Sauerbrey, R., Bussmann, M.: Picongpu: a fully relativistic particle-in-cell code for a gpu cluster. IEEE Transactions on Plasma Science **38**(10), 2831–2839 (2010)
5. Byna, S., Sisneros, R., Chadalavada, K., Koziol, Q.: Tuning parallel i/o on blue waters for writing 10 trillion particles. Cray User Group (CUG) (2015)
6. Byna, S., Chou, J., Rubel, O., Karimabadi, H., Daughter, W.S., Roytershteyn, V., Bethel, E.W., Howison, M., Hsu, K.J., Lin, K.W., et al.: Parallel I/O, analysis, and visualization of a trillion particle simulation. In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 1–12. IEEE (2012)
7. Catrina, O., Saxena, A.: Secure computation with fixed-point numbers. In: International Conference on Financial Cryptography and Data Security. pp. 35–50. Springer (2010)
8. Chandrasekaran, S., Debus, A., Kluge, T., Widera, R., Bastrakov, S., Steiniger, K., Garten, M., Werner, M., Kelling, J., Leinhauser, M., Young, J., Davis, J.H., Diaz, J.M., Worpitz, B., Huebl, A.: Running picongpu on summit: Caar: Preparing picongpu for frontier at ornl. In: 4th OpenPOWER Academia Discussion Group Workshop (2019)
9. Chen, G., Chacón, L., Yin, L., Albright, B.J., Stark, D.J., Bird, R.F.: A semi-implicit, energy-and charge-conserving particle-in-cell algorithm for the relativistic vlasov-maxwell equations. Journal of Computational Physics **407**, 109228 (2020)
10. Choquette, J., Gandhi, W.: Nvidia A100 GPU: Performance & innovation for GPU computing. In: 2020 IEEE Hot Chips 32 Symposium (HCS). pp. 1–43. IEEE Computer Society (2020)
11. Dawson, J.M.: Particle simulation of plasmas. Reviews of modern physics **55**(2), 403 (1983)
12. Edwards, H.C., Trott, C.R., Sunderland, D.: Kokkos: Enabling many-core performance portability through polymorphic memory access patterns. Journal of Parallel and Distributed Computing **74**(12), 3202 – 3216 (2014). https://doi.org/https://doi.org/10.1016/j.jpdc.2014.07.003, http://www.sciencedirect.com/science/article/pii/S0743731514001257, domain-Specific Languages and High-Level Frameworks for High-Performance Computing
13. Fonseca, R.A., Silva, L.O., Tsung, F.S., Decyk, V.K., Lu, W., Ren, C., Mori, W.B., Deng, S., Lee, S., Katsouleas, T., et al.: Osiris: A three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators. In: International Conference on Computational Science. pp. 342–351. Springer (2002)

14. Fonseca, R.A., Vieira, J., Fiúza, F., Davidson, A., Tsung, F.S., Mori, W.B., Silva, L.O.: Exploiting multi-scale parallelism for large scale numerical modelling of laser wakefield accelerators. Plasma Physics and Controlled Fusion **55**(12), 124011 (2013)
15. Fried, B.D.: Mechanism for instability of transverse plasma waves. Physics of Fluids **2**(3), 337–337 (1959)
16. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM computing surveys (CSUR) **23**(1), 5–48 (1991)
17. Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D.T., Jammalamadaka, N., Huang, J., Yuen, H., et al.: A study of bfloat16 for deep learning training. arXiv preprint arXiv:1905.12322 (2019)
18. Li, A., Song, S.L., Chen, J., Li, J., Liu, X., Tallent, N.R., Barker, K.J.: Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect. IEEE Transactions on Parallel and Distributed Systems **31**(1), 94–110 (2019)
19. Morse, R., Nielson, C.: Numerical simulation of the weibel instability in one and two dimensions. The Physics of Fluids **14**(4), 830–840 (1971)
20. NVIDIA Corporation: Nvidia A100 tensor core GPU architecture. Tech. rep. (2020), https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf
21. Stix, T.H.: Waves in plasmas. Springer Science & Business Media (1992)
22. Thode, L., Sudan, R.: Two-stream instability heating of plasmas by relativistic electron beams. Physical Review Letters **30**(16), 732 (1973)
23. Vay, J.L., Almgren, A., Bell, J., Ge, L., Grote, D., Hogan, M., Kononenko, O., Lehe, R., Myers, A., Ng, C., et al.: Warp-x: A new exascale computing platform for beam–plasma simulations. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **909**, 476–479 (2018)