Optimizing Vector Particle-In-Cell (VPIC) for Memory Constrained Systems Using Half-Precision

Nigel Tan[1], Robert Bird[2] (advisor), Michela Taufer[1] (advisor)
[1]University of Tennessee Knoxville, [2]Los Alamos National Laboratory

Contributions:

Develop optimizations to VPICs particle storage format that reduces particle memory usage by up to 31.25% and enables an increase in particle count by up to 40%

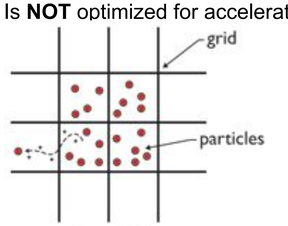
Demonstrate that our optimizations enable significantly larger simulations and produce accurate scientific results

Motivation

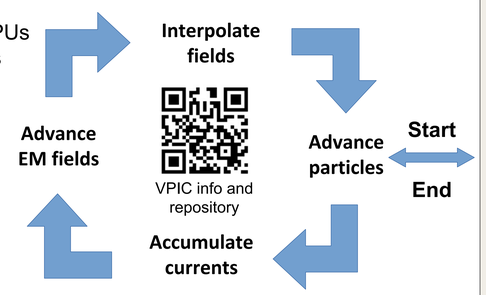
- Particle simulations require vast quantities of particles to model real world phenomenon with modern simulations reaching trillions of particles [1]
- **Particle simulation scale is more limited by memory than compute power**
- Memory growth cannot keep up with compute growth
- Systems are shifting more compute power into accelerators which further limits scale
  - Modern CPUs support up to 4 TB of memory
  - GPUs are limited to at most 32 GB
  - Data movement between CPU and accelerators is costly (PCIe 4.0 BW: 32 GB/s)

Vector Particle-In-Cell (VPIC)

- High performance 3D PIC code developed by Los Alamos National Laboratory with a long history of large scale simulations
  - Simulate magnetic reconnection, fusion, solar weather, amongst other plasma phenomenon
  - Is highly optimized for modern CPUs
  - Is **NOT** optimized for accelerators



Spatial domain: Particles are distributed across an n-D space that is decomposed into a n-D grid



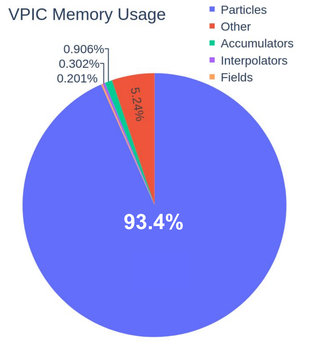
Iterative process: Four key steps define a VPIC iteration

Particle Representation and Storage

- Each particle requires 32 bytes as shown below
- Particles take up >90% of memory usage
- Cell index representation is already optimal and momentum is difficult to shrink

VPIC Particle Structure = 1 byte

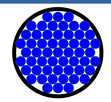
```
float dx // Position
float dy
float dz
float ux // Momentum
float uy
float uz
int   i  // Cell index
float w  // Weight
```



Memory usage of key VPIC structures using a test problem with 317,030,400 particles.

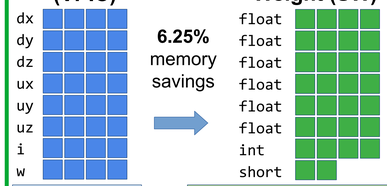
Optimizing Particle Weight Storage

- Each simulated particle is a macroparticle
- Weight defines how many real particles modeled by the macroparticle
- Particle weight generally **does not change**



Example of macroparticle modeling 55 real particles

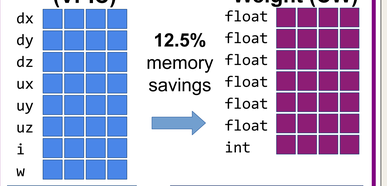
Default (VPIC) → **Short Weight (SW)**: **6.25%** memory savings



Each particle stores its own weight as float

Store weight as multiple of a known base value unique to each species of particles

Default (VPIC) → **Constant Weight (CW)**: **12.5%** memory savings



Each particle stores its own weight as float

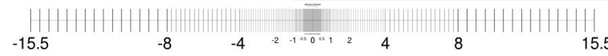
Use a constant weight for all particles of the same species

Half Precision Particle Position Storage

Compared to Half-Precision

- Bfloat16 lose precision (decimal digits)
- TensorFloat requires more storage (32 bits vs 16 bits)

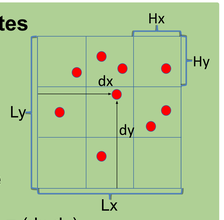
Precision	Sign	Exponent	Fraction	Decimal Digits
Double	1	11	52	~15.9
Single	1	8	23	~7.2
Half	1	5	10	~3.3
~~Bfloat16~~	~~1~~	~~8~~	~~7~~	~~2.4~~
~~TensorFloat~~	~~1~~	~~8~~	~~10~~	~~3.3~~



IEEE 754 Floating-point numbers are **"logarithmically" distributed** [2]

Global Coordinates

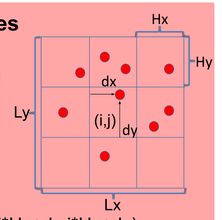
- Points further away from 0 lose precision
- Wastes bits representing points outside the grid space



Position = (dx,dy)

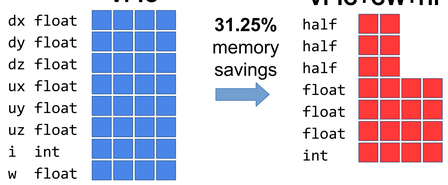
Local Coordinates

- Points are more evenly distributed
- Enable lower precision for position on a sufficiently fine resolution grid



Position = (i*Hx+dx,j*Hy+dy)

- Half-precision position **(HP)** combined with constant particle weight **(CW)** drastically reduces storage requirements and allows more particles to fit in the same amount of memory

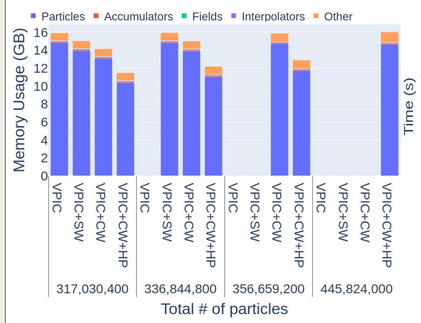


Results

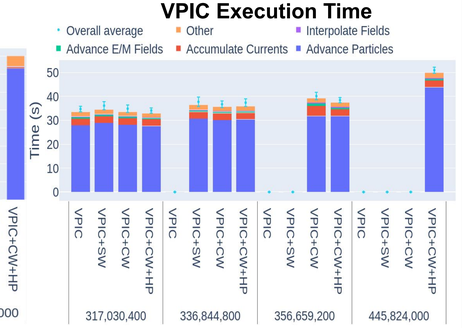
- Experiments conducted on a Power9+V100 system
- Targeting Nvidia GPUs due to CUDA having the most complete and convenient half-precision support
  - * Operate on 128/256 bit registers only
  - ** Support varies on implementation

Architecture	FP16 Storage	FP16 Conversion	FP16 Arithmetic
Intel x86-64	✓	*	X
AMD x86-64	✓	*	X
ARMv8.2-A	✓	✓	**
IBM Power9	✓	*	X
Nvidia GPUs	✓	✓	✓
AMD GPUs	✓	✓	✓

VPIC Memory Usage

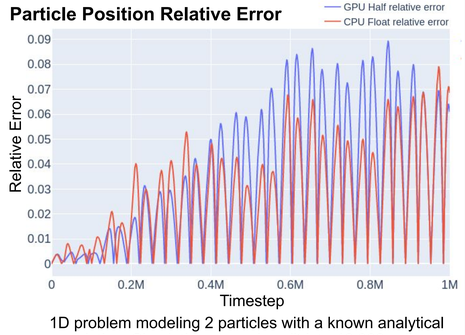


VPIC Execution Time



- Memory usage and runtime performance tests model laser-plasma interactions
- Using half-precision for particle position combined with constant weight particles enables a **40.625%** increase in particle count
- Optimized weight and position impose minimal performance impact
- For a sufficiently high resolution grid, half-precision achieves similar accuracy to original VPIC

Particle Position Relative Error



1D problem modeling 2 particles with a known analytical solution. Simulation space is split into 10,000 cells.

Conclusions

- Half precision combined with local coordinates is a viable path for reducing particle storage while maintaining accuracy
- Optimized particle storage enables a ~40% increase in number of particles simulated using the same amount of memory

Future Work

- Add support for CPU half precision hardware
- Increase precision by using fixed point particle position
- Investigate optimizations for particle momentum

References

1. Byna, Suren, et al. "Tuning parallel i/o on blue waters for writing 10 trillion particles." Cray User Group (CUG) (2015).
2. https://www.volkerschatz.com/science/float.html

Acknowledgements

Work performed under the auspices of the U.S. Department of Energy by the Triad National Security, LLC Los Alamos National Laboratory for the DOE's National Nuclear Security Administration (Contract No. 89233218CNA000001). Support provided by the Advanced Simulation and Computing Program.

LA-UR-20-25988

GCLab information