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

Student: Nigel Tan

Mentors: Robert Bird, Michela Taufer

GCLab
UNIVERSITY OF TENNESSEE

THE UNIVERSITY OF TENNESSEE KNOXVILLE
BIG ORANGE. BIG IDEAS.®

Los Alamos NATIONAL LABORATORY
EST. 1943

Scaling Particle Simulations

0.374 Pflop/s Trillion-Particle Kinetic Modeling of Laser Plasma Interaction on Roadrunner

K. J. Bowers, *Member, IEEE*, B. J. Albright, *Member, IEEE*, B. Bergen, *Member, IEEE*, L. Yin, K. J. Barker and D. J. Kerbyson, *Member, IEEE*

2 trillion particles = 64 TB
(2008)

Tuning Parallel I/O on Blue Waters for Writing 10 Trillion Particles

Surendra Byna*, Robert Sisneros†, Kalyana Chadalavada†, and Quincey Koziol‡

10 trillion particles = 320 TB
(2015)

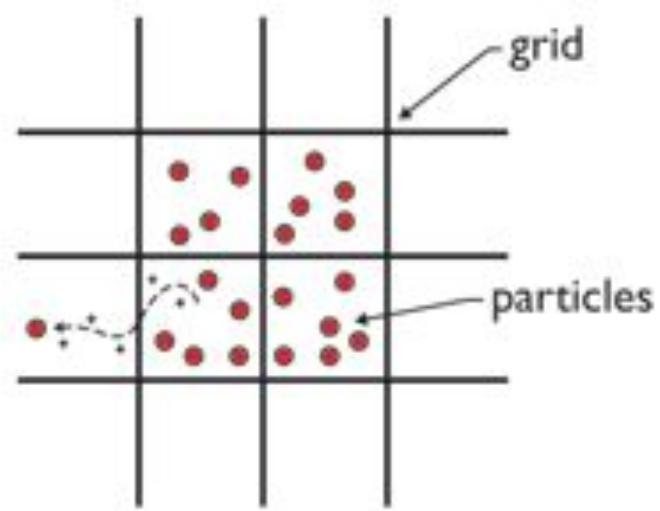
- Simulation scale more limited by memory than compute
- Accelerators add more memory constraints
  - Max CPU memory: **4TB**, Max GPU memory: **48GB**
  - PCIe 4.0 x16 Bandwidth: **32 GB/s** in one direction

(left) Bowers, Kevin J., et al. "0.374 Pflop/s trillion-particle kinetic modeling of laser plasma interaction on Roadrunner." SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. IEEE, 2008.
(right) Byna, Suren, et al. "Tuning parallel i/o on blue waters for writing 10 trillion particles." Cray User Group (CUG) (2015).

Vector Particle-In-Cell (VPIC)

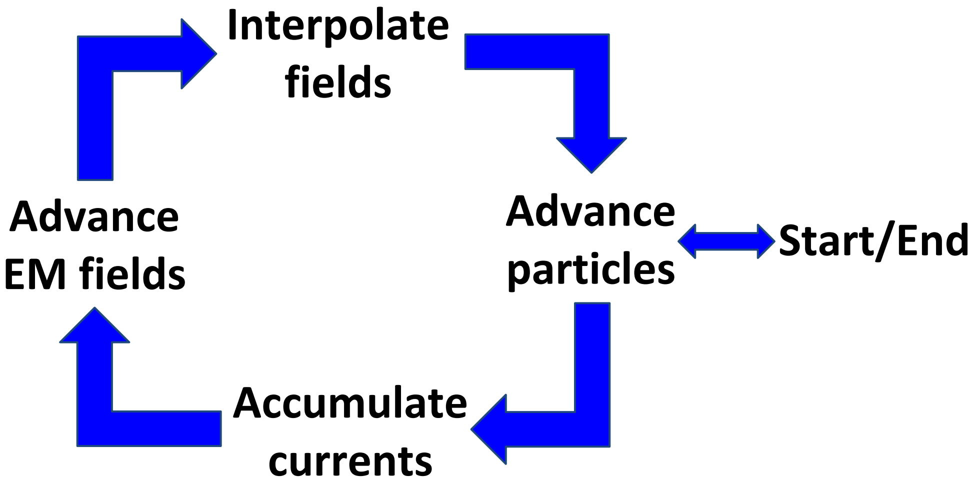
- High performance particle-in-cell code for plasma simulations:
  - Simulates magnetic reconnection, fusion, solar weather, and particle acceleration amongst other plasma phenomenon
  - One of the fastest plasma codes in the world
  - Is well optimized for modern CPUs
  - Was **NOT** optimized for accelerators (e.g., GPUs)

VPIC Algorithm



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

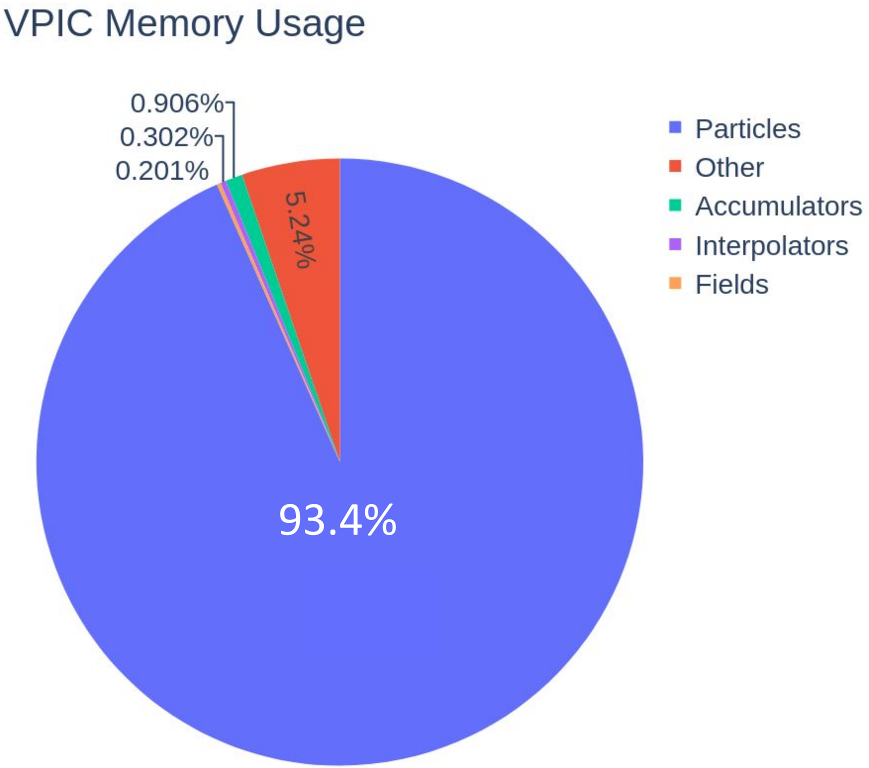


BIG ORANGE BIG IDEAS

Particle Storage

- The larger the number of particles, the more physically accurate the simulations and the greater the memory usage

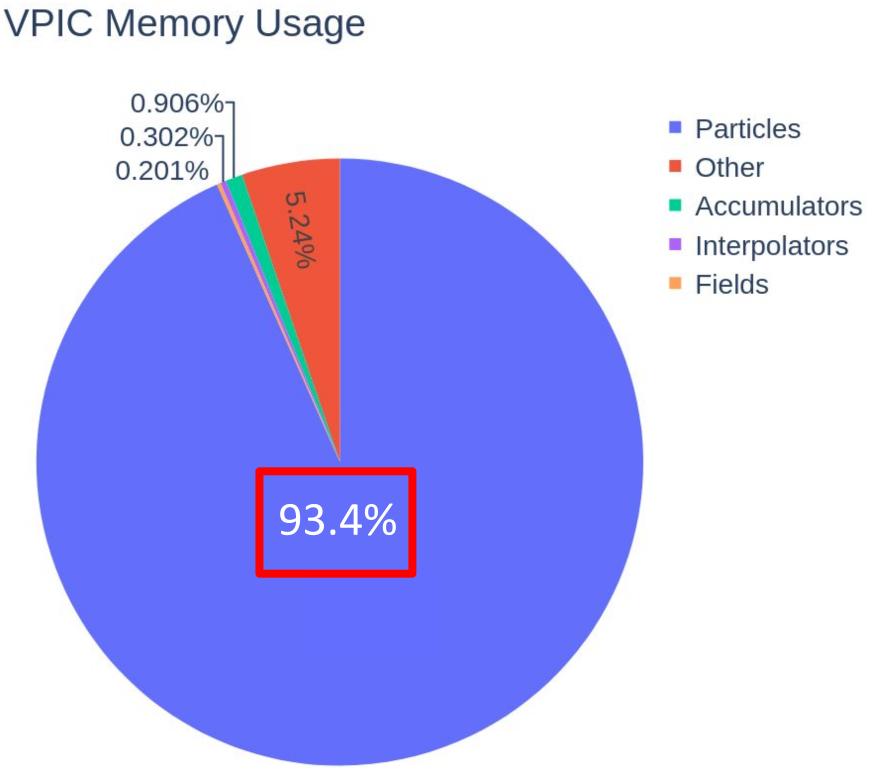
```
struct particle {
  float dx, dy, dz; // Position
  int i;            // Cell index
  float ux, uy, uz; // Momentum
  float w;          // Weight
};
```



Particle Storage

- The larger the number of particles, the more physically accurate the simulations and the greater the memory usage

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

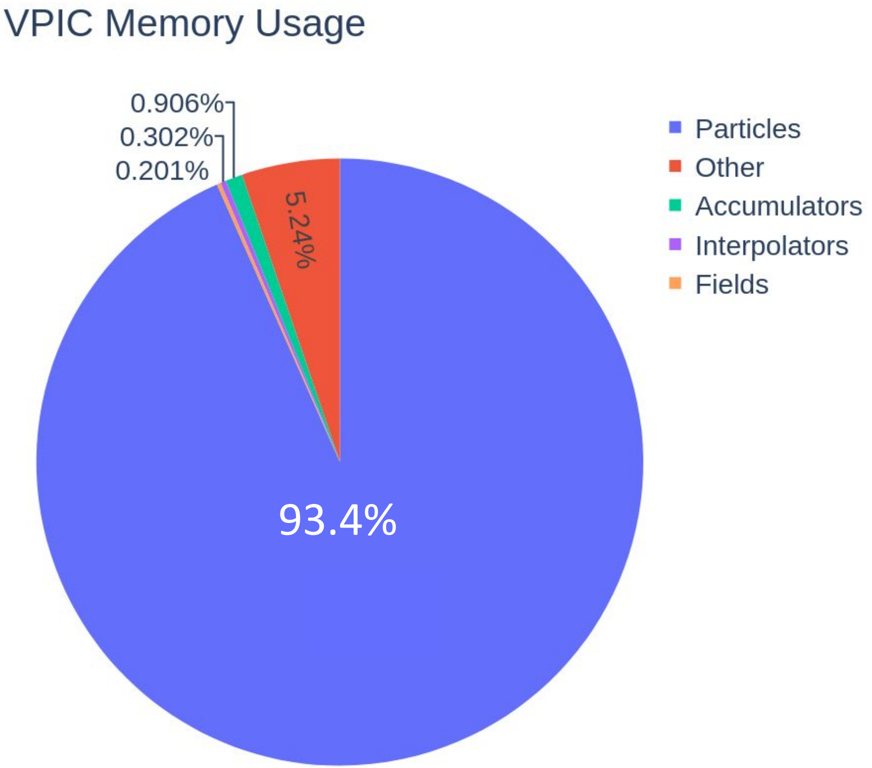


BIG ORANGE BIG IDEAS

Particle Storage: Weight

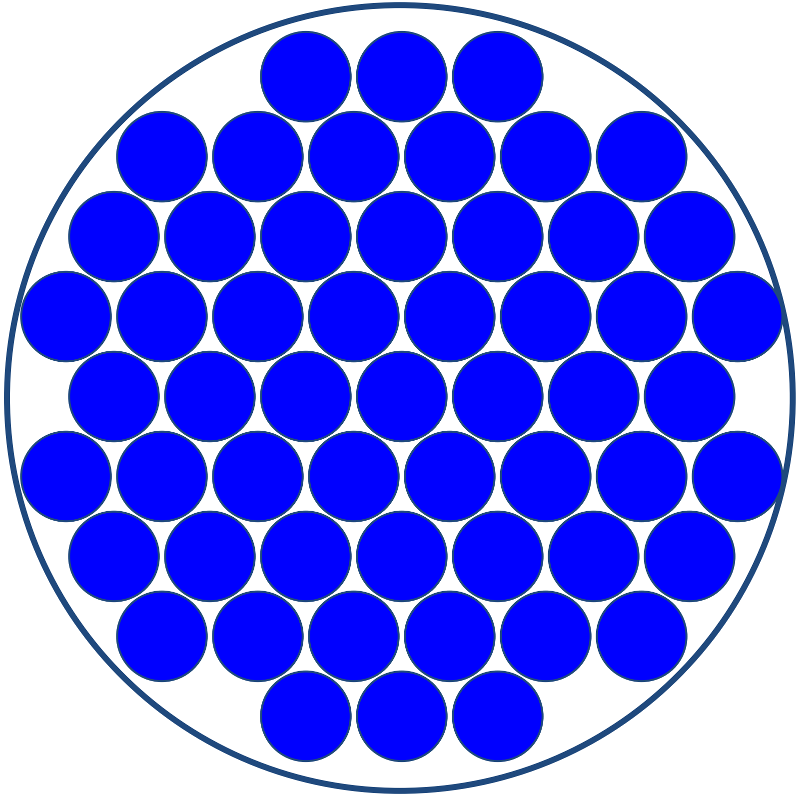
- The larger the number of particles, the more physically accurate the simulations and the greater the memory usage

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



Particle Weight

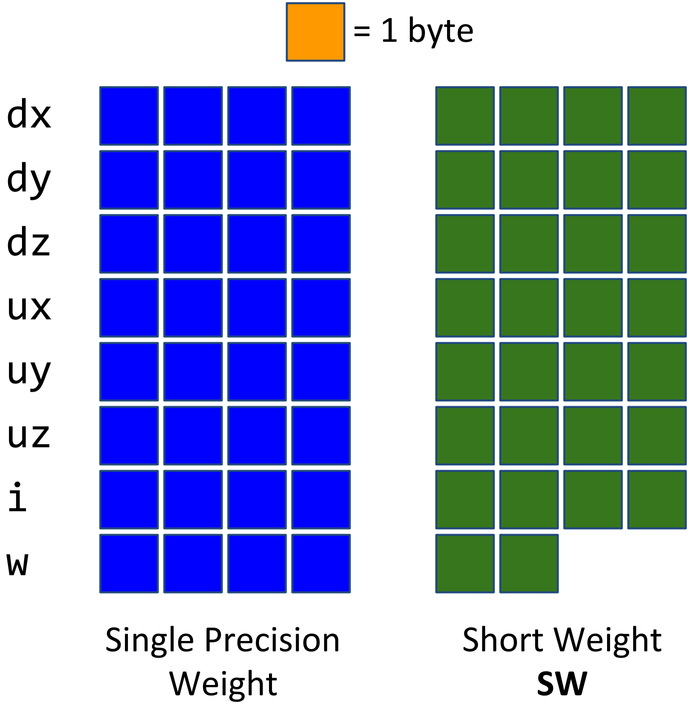
- Each simulated particle is a macroparticle
- Weight defines the number of real particles modeled by each macroparticle
- **Weight generally does not change during a simulation**



Example of macroparticle modeling
55 real particles

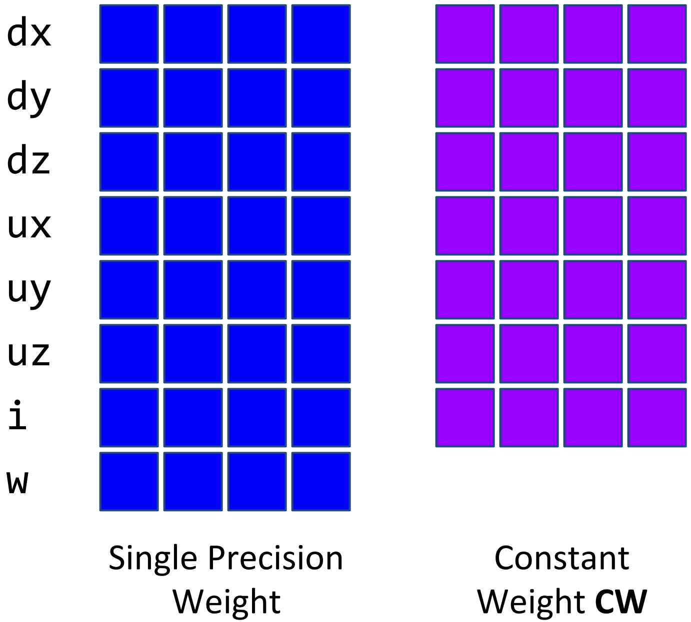
Optimizing Particle Weight Storage (I)

- Assume particle weights may vary but have a limited range of values
  - Weights have a common divisor
- Replace weight with 16-bit short integer **(SW)**
- Reduce particle memory usage by at most **6.25%** over default VPIC



Optimizing Particle Weight Storage (II)

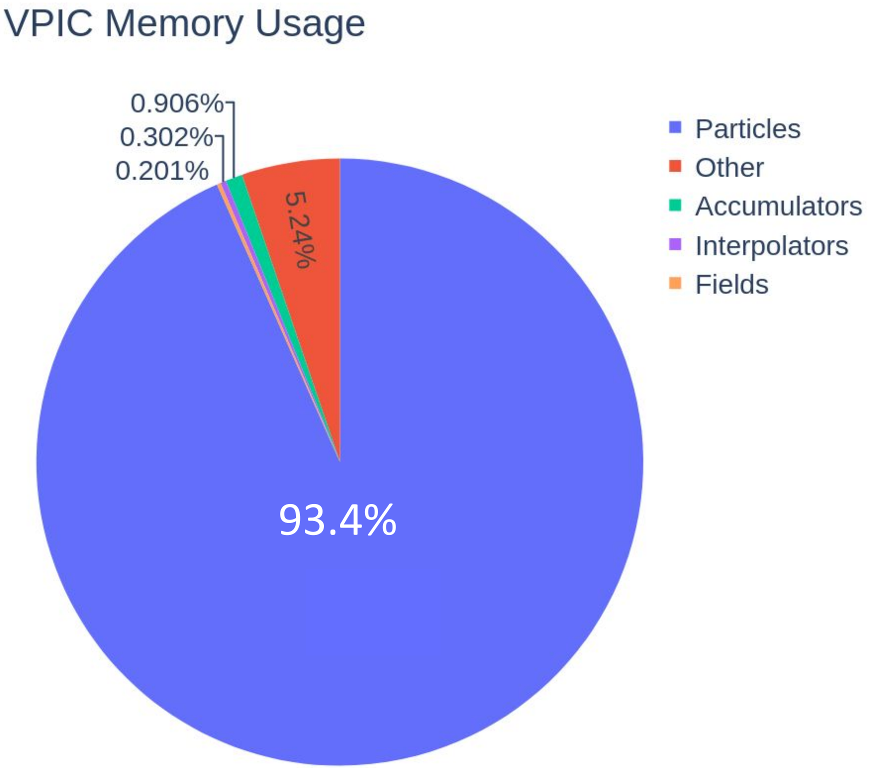
- Assume all particles in a species share the same constant weight **(CW)**
- Remove weight field and use a per species constant weight
- Reduce particle storage cost by at most **12.5%** over default VPIC



Particle Storage: Position

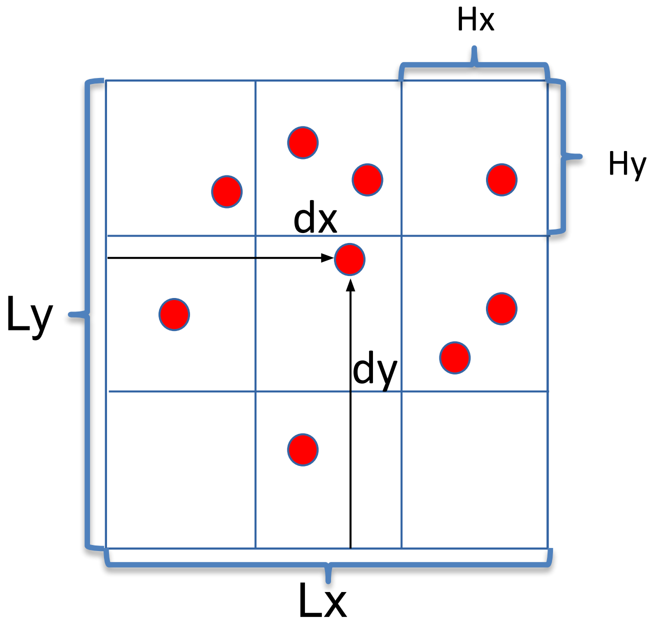
- The larger the number of particles, the more physically accurate the simulations and the greater the memory usage

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



Particle Position: Global Coordinates

- Traditional global coordinates:
  - Derive cell index based on global position
- **Problem:** Uneven floating point intervals
  - Coordinates of particles away from 0.0 are less precise

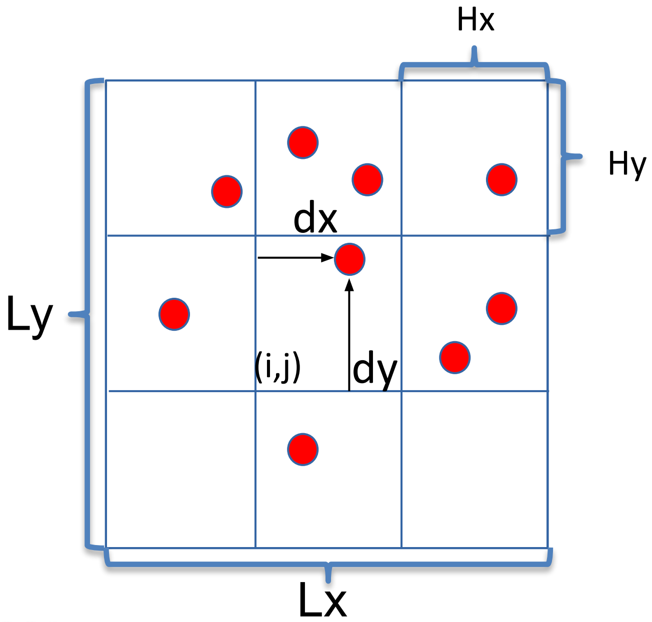


Particle position: (dx, dy)

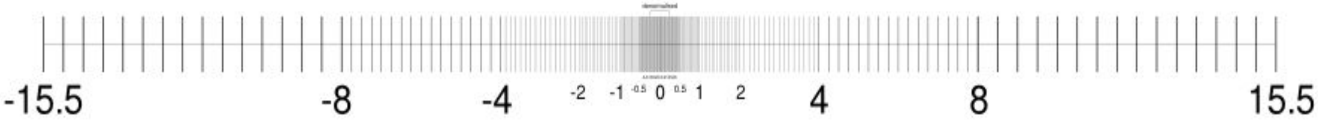
https://www.volkerschatz.com/science/float.html

Particle Position: Local Coordinates

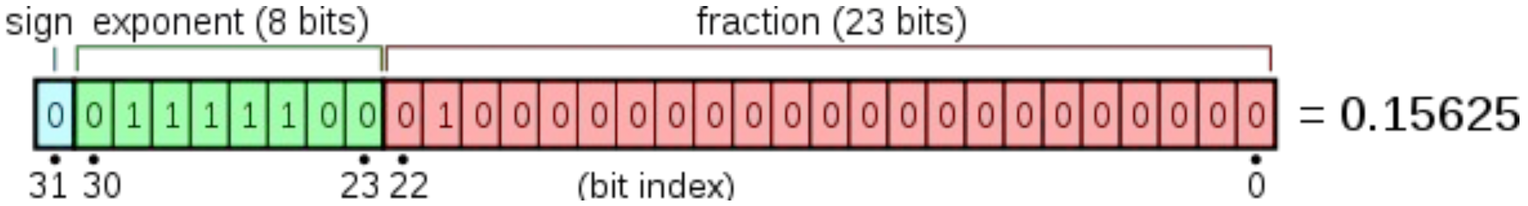
- Local coordinates:
  - Derive particle position based on cell index and position within a cell
- **Advantages:** More floating-point values that are more evenly dispersed
  - Enable lower precision with similar accuracy on a high resolution grid



Particle position:
(i*Hx+dx, j*Hy+dy)



Floating Point Format



Compared to FP16:

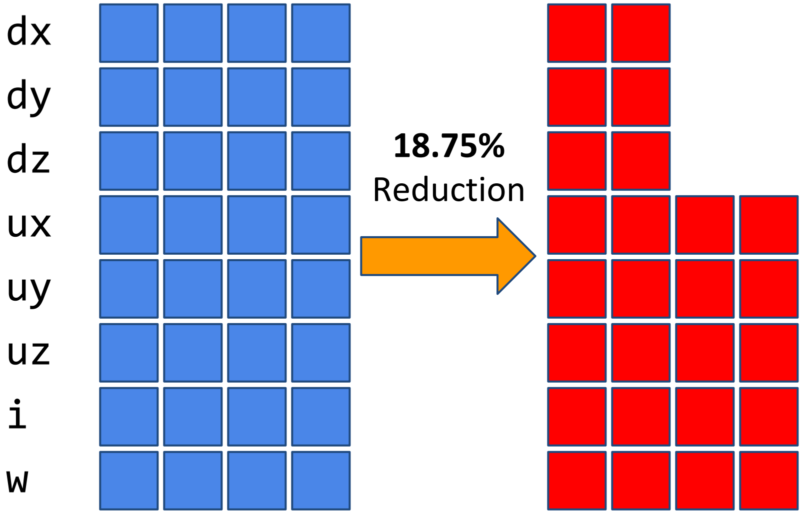
- Bfloat16 lose precision (decimal digits)
- TensorFloat requires more storage (19 bits)

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

BIG ORANGE BIG IDEAS

Half-Precision Particle Position

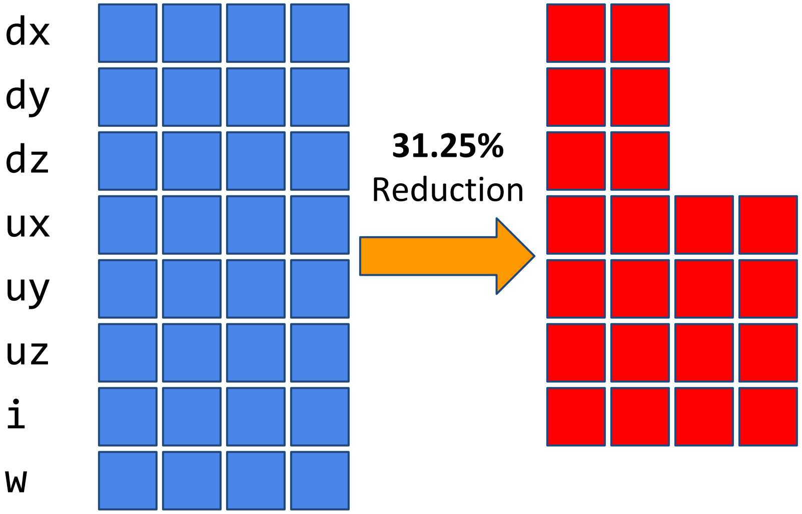
Half Precision Position



Single Precision Position

Half Precision Position **(HP)**

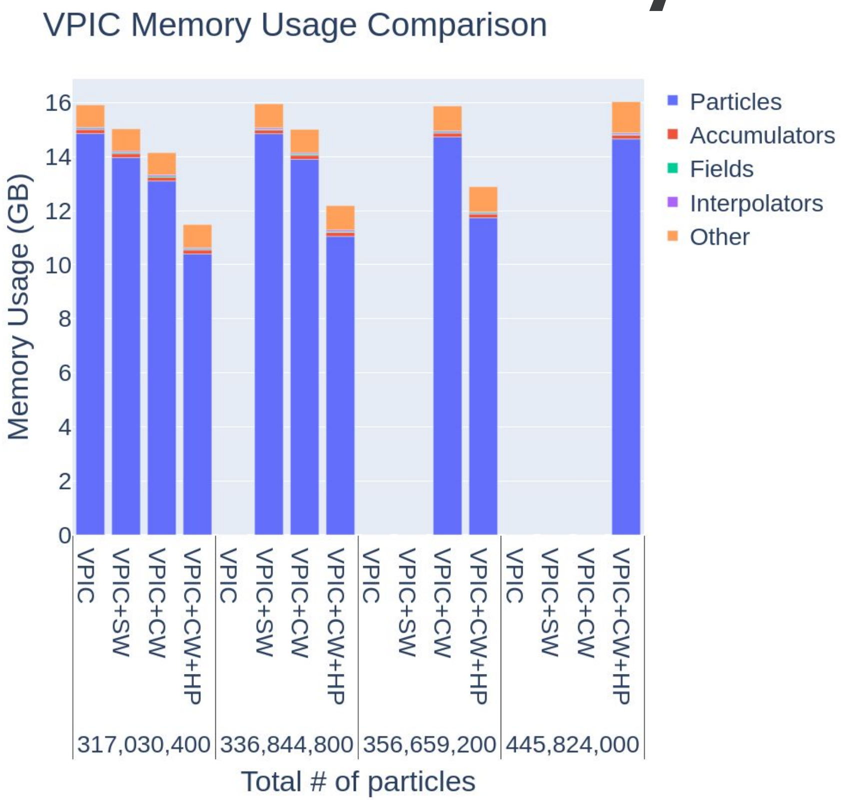
Half Precision Position + Constant Weight



Single Precision Position

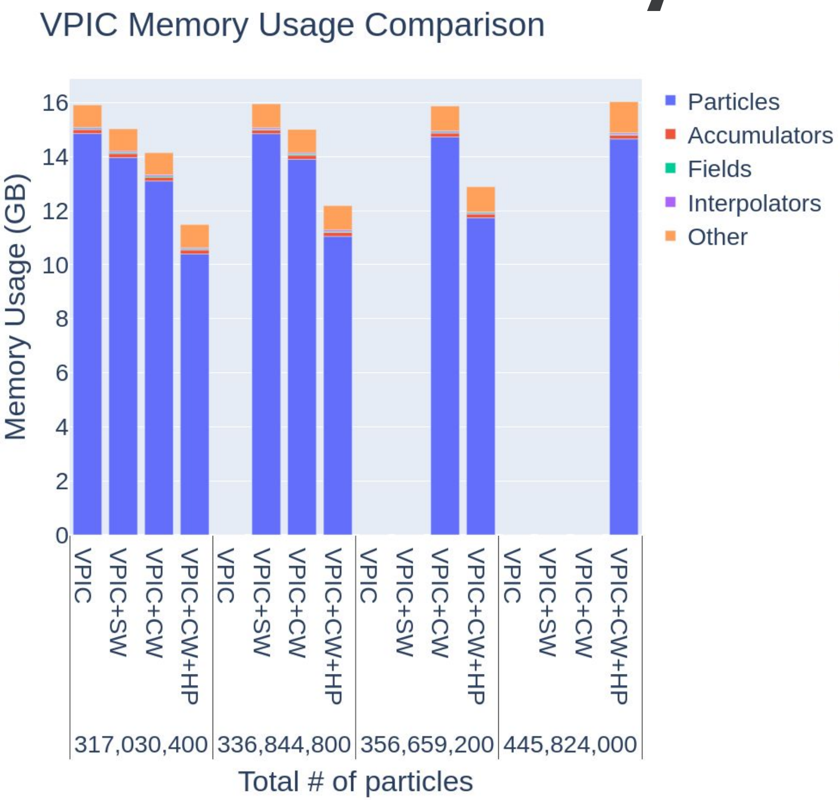
Half Precision Position **(HP + CW)**

Results: Memory Usage and Runtime



- Laser-Plasma interaction simulation
- Missing bars indicate out of memory
- Optimizations enable up to 40% increase in number of particles

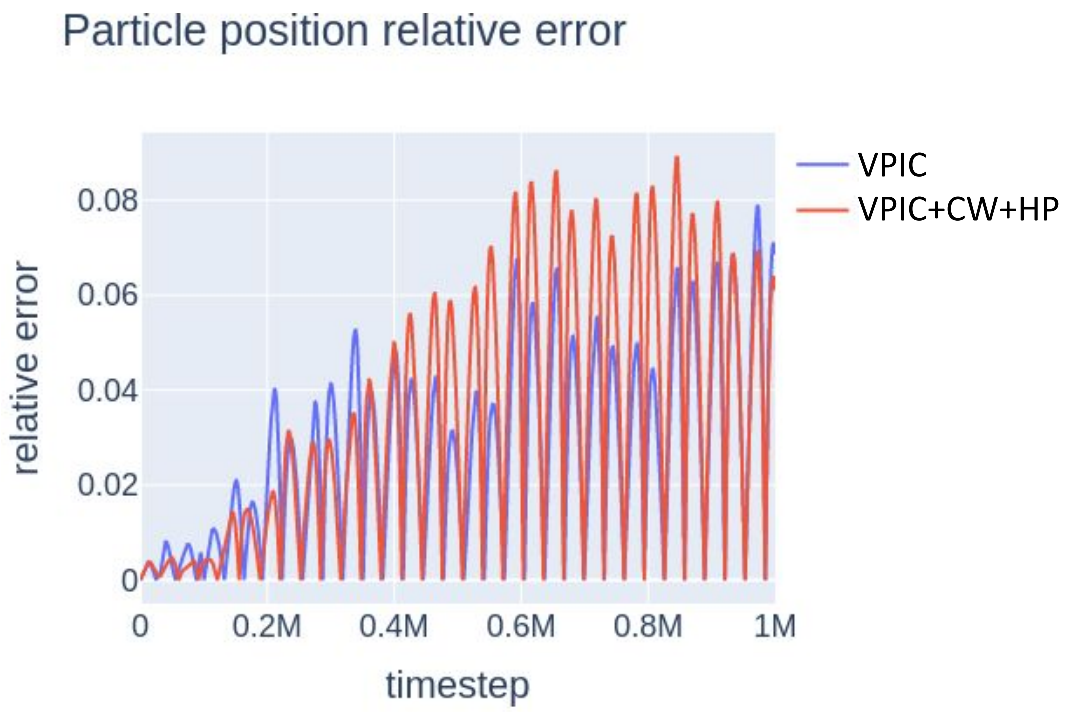
Results: Memory Usage and Runtime



BIG ORANGE BIG IDEAS

Results: Accuracy

- Perform as well as the original single precision VPIC with a sufficiently fine grid
- Weight kept constant and does not affect overall accuracy



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

Conclusions

- Our optimizations enable an up to 40% increase in particle count
- Maintain VPICs high performance
- Produce scientifically accurate results
- Demonstrate the potential in lower precision storage in scientific applications

Next Steps

- Add half-precision support for CPUs
- Investigate alternative formats for position
- Develop model for automatically determining whether to use half-precision based on simulation settings
- Develop optimizations for particle momentum