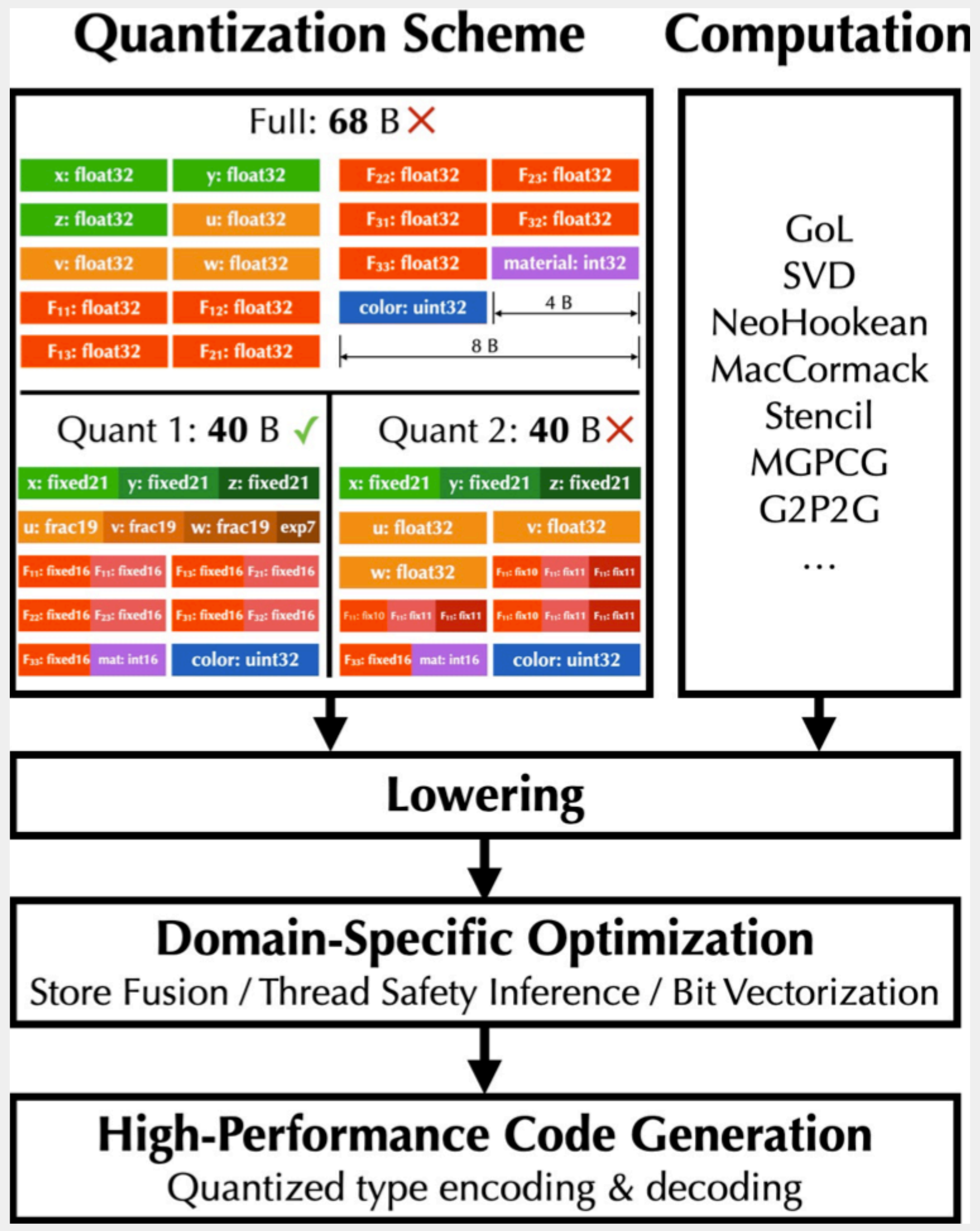
QuanTaichi: A Compiler for Quantized Simulations

Yuanming Hu[1,2], Jiafeng Liu[3], Xuanda Yang[3], Mingkuan Xu[1,4], Ye Kuang[1], Weiwei Xu[3], Qiang Dai[5], William T. Freeman[2], Fredo Durand[2]

[1]Taichi Graphics [2]MIT CSAIL [3]State Key Laboratory of CAD&CG, Zhejiang University
[4]Tsinghua University [5]Kuaishou Technology

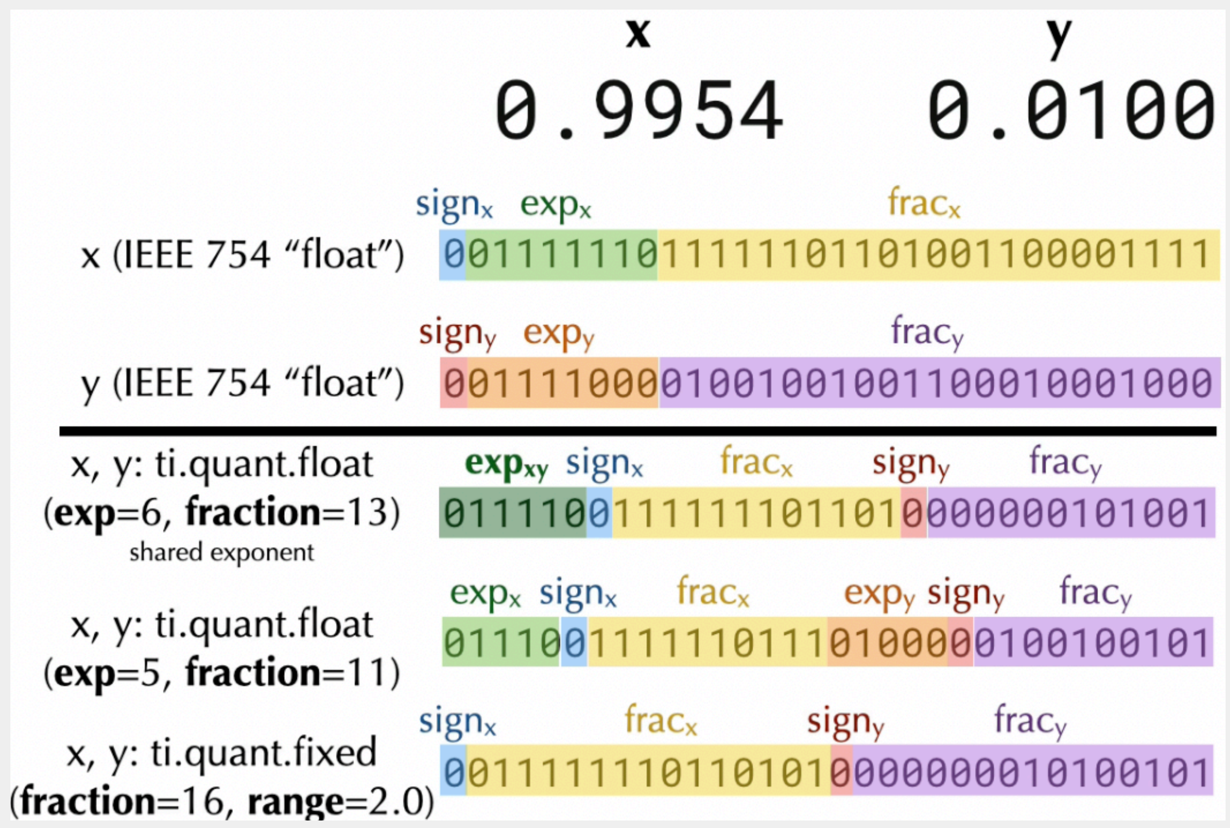
Introduction

- High-resolution simulations can deliver great visual quality, but they are often limited by available memory, especially on GPUs.
- We present a compiler for physical simulation that can achieve both high performance and significantly reduced memory costs, by enabling flexible and aggressive quantization.



Key ideas

- Use bit-level compression to save memory!
- An example: packing a 2D coordinate (`x` and `y`) into 32 bits.



Methods

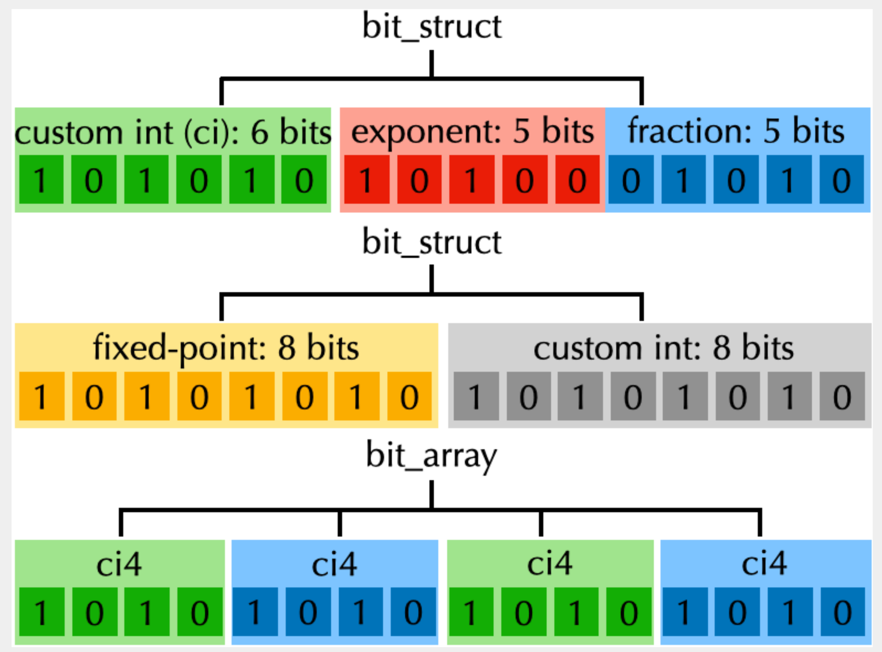
To realize quantized simulations, we need to implement new type systems.

- Custom numerical data types:

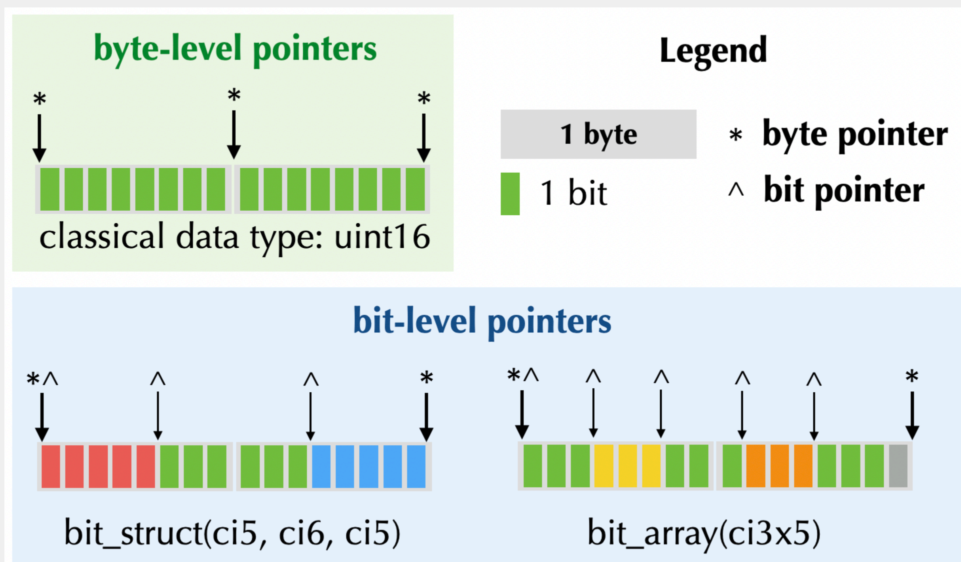
  We support 3 types of custom data types: custom integer, fixed-point numbers and floating-point numbers

```python
# custom integer
i5 = ti.quant.int(bits=5)
# fixed-point real numer
fixed17 = ti.quant.fixed(frac=17, range=3.14)
# floating-point real numer
f18 = ti.quant.float(exp=4, frac=14)
```

- Bit adapters
  - We use bit adapters to pack custom data types into hardware-native types
  - Bit structs: structs of different custom types.
  - Bit arrays: arrays of repeated custom types.



- Bit levels pointers
  - Bit level pointers are `traditional byte pointers + bit offset`.
  - With bit pointers we can locate each single bit.

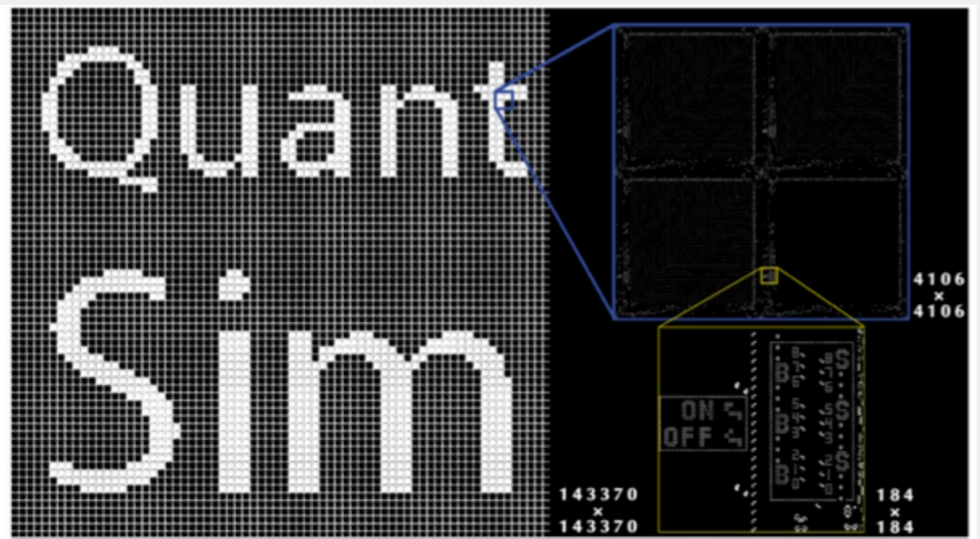


- Domain specific optimizations
  - Some of the operations involved in quantized simulations could be quite costly, so we propose the following domain specific optimizations for performance:
    - Bit struct store fusion
    - Thread safety inference
    - Bit array vectorization

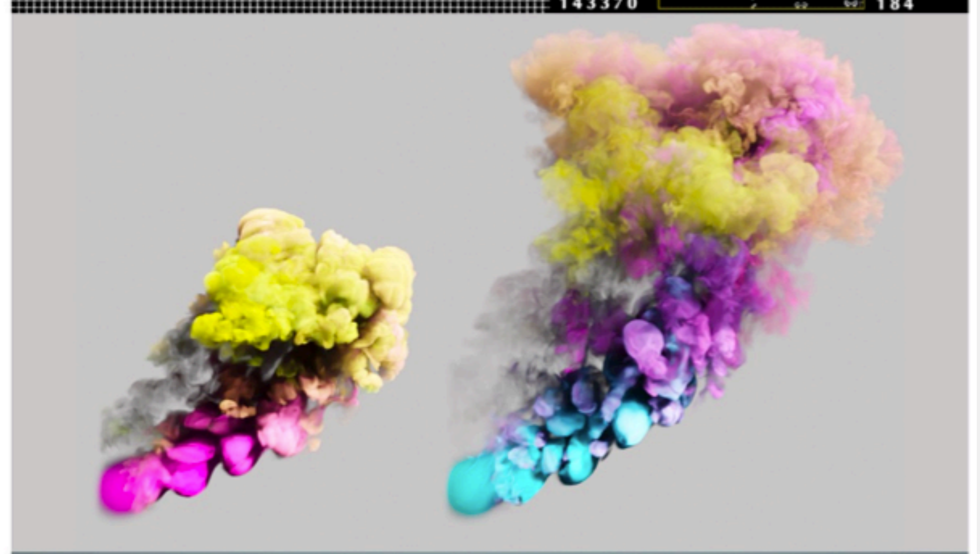
Results

- With our compiler we implement the following three, to our knowledge, largest scale simulations on a single GPU respectively.

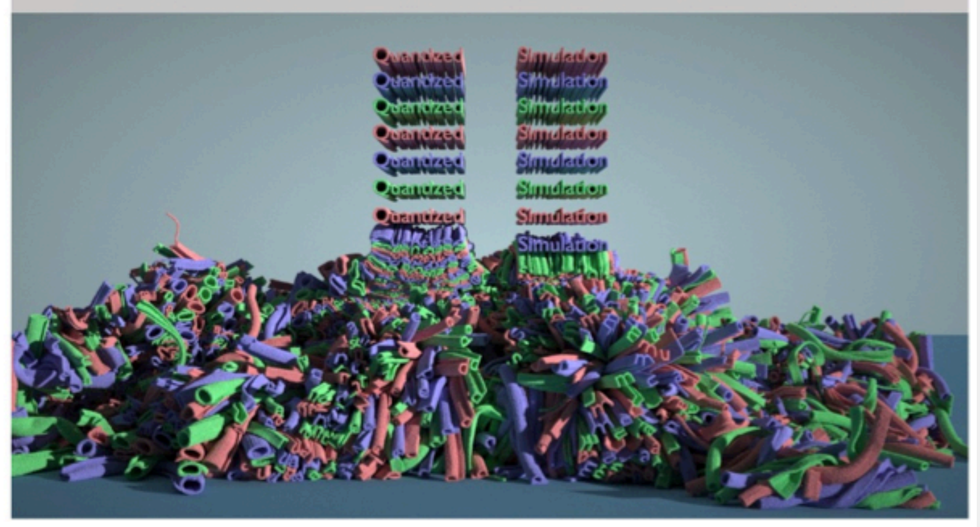
Game of Life
7.0 GB memory
20,554,956,900 cells
Per cell: 2 B→0.25 B
(8.0x)



Advection-Reflection
29.3 GB memory
421,134,336 voxels
Per voxel: 110 B→70 B
(1.6x)



MLS-MPM
16.6 GB memory
234,527,481 particles
Per particle: 68B→40B
(1.7x)



- Performance
  - Thanks to the domain specific optimizations and memory bandwidth saving, the performance of the quantized simulators is comparable and even better the non-quantized simulators in our experiments.
  - Please check our paper for more details.

Conclusions

- Memory saving: users can save memory(1.6x ~ 8x in our experiments) with our compiler.
- Easy to use. No more than 3% LoC modification can make a traditional simulator quantized.
- The performance is comparable, sometimes even better than full-precision version.
- No significant visual quality degradation (more details in paper)