LLNL-POST-728998 This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

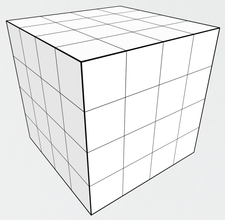
Reducing Data Movement using Adaptive-Rate Computing

P. Lindstrom (LLNL)

Data movement is becoming the primary performance bottleneck in high-performance computing. Proposed mitigation strategies include reducing the precision of data stored and moved. For instance, mixed-precision arithmetic performs some computations in single or half precision at the expense of accuracy, while inline data compression uses reduced but fixed-size storage, with precision dictated by how well the data compresses. Such uniform fixed-rate compression uses enough storage to ensure minimum precision in active and numerically sensitive regions of a computational domain, but wastes precious bits on quiescent regions. We are developing adaptive-rate compressed arrays that vary storage size spatially to reduce data movement while satisfying user-specified precision requirements.

ZFP: Compressed floating-point arrays significantly reduce memory usage

Our ZFP compressor stores floating-point arrays compressed. The *d*-dimensional arrays are partitioned into blocks of 4^d values that are independently compressed and decompressed on demand at very high speed (up to 2 GB/s/core).



A 3D ZFP block consists of 4×4×4 floating-point values. These values are compressed to a progressive, variable-length bit string that can be truncated at any point. The more bits that are retained, the higher the numerical accuracy, much like how single-precision floats can be thought of as truncated double-precision values.

ZFP supports several compression modes that can be independently selected for each individual block, if so desired:

- **Fixed rate**: Uniform **storage size** enables fast random access.
- **Fixed precision**: Uniform **relative error** using variable-size blocks.
- **Fixed accuracy**: Uniform **absolute error** using variable-size blocks.

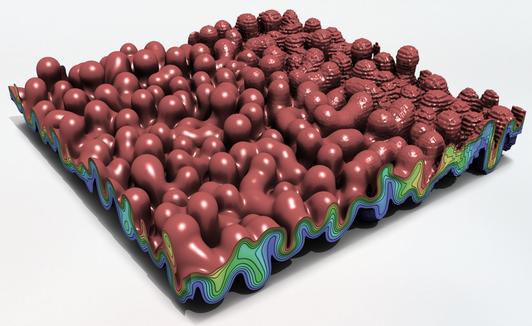
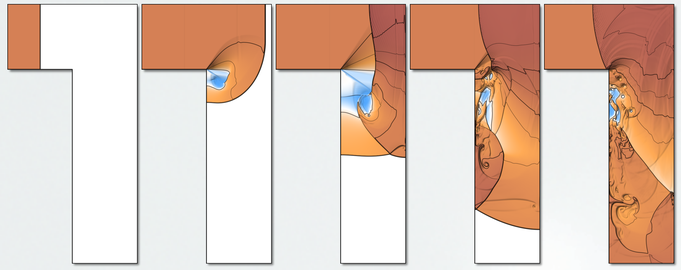


Illustration of varying the rate spatially over the domain, from 4 bits/value on the left to less than half a bit/value on the right, representing as much as 170:1 compression over IEEE double precision.

ZFP arrays are available as C++ classes that use operator overloading to hide from the user the details of compression, decompression, and caching, allowing them to replace conventional uncompressed arrays with minimal code changes. These arrays support random read/write access via ZFP's fixed-rate mode.

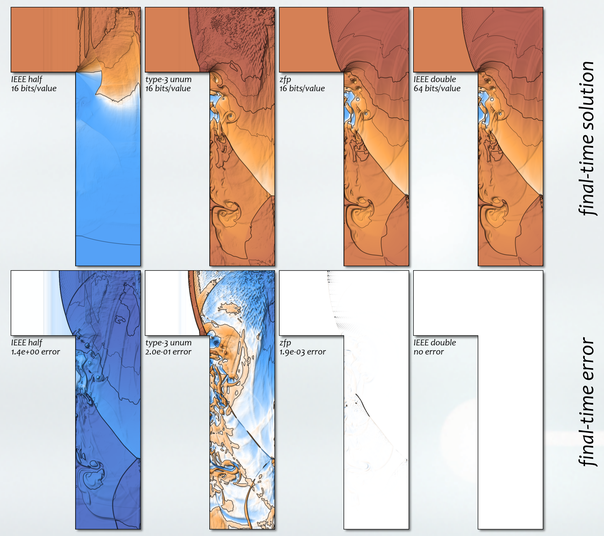
ZFP fixed-rate storage increases accuracy over IEEE by three orders of magnitude

Conventional simulation codes use 64-bit double-precision floating point storage to represent state variables. To evaluate the effectiveness of reduced precision representations, we use an Eulerian finite-volume shock propagation code.



Snapshots in time of a shock wave propagating through an L-shaped chamber.

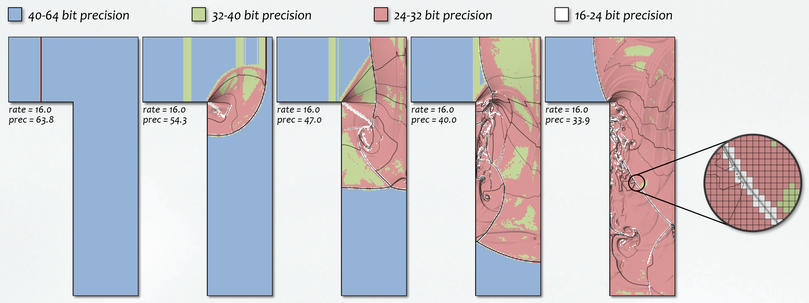
Comparisons with 16-bit representations—IEEE half precision, type-3 unums proposed by John Gustafson, and ZFP—reveal that ZFP gives an almost identical solution to the "ground truth" 64-bit solution, but using four times less storage.



Top row: Final-time solution given by IEEE half precision, type-3 unums, ZFP, and full IEEE double precision. Note how the shock dissipates in the half-precision solution, while ZFP closely matches the true solution. Bottom row: Visualization of the error with respect to IEEE double precision.

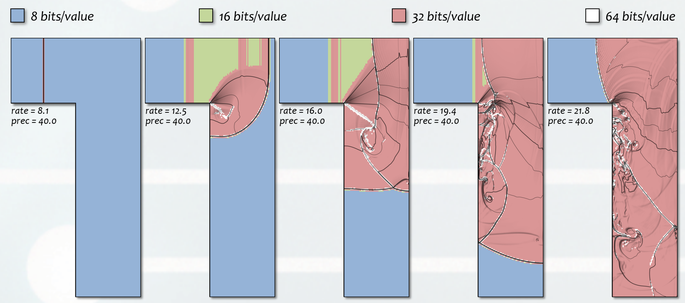
Adaptive-rate computing ensures bits are allocated to where they are most needed

A uniform rate allocation over the domain has two downsides. First, too many bits are spent on easy-to-compress regions, resulting in an unnecessarily precise representation of the wake of the shock and the "background" ahead of it, which have little impact on the evolution of the simulation. Conversely, the active regions near the shock waves are difficult to compress and are therefore represented using far less precision, limiting solution accuracy.



Visualization of how the numerical precision varies spatially when fixing the rate at 16 bits/value. Note in particular that the most active and important regions (white) around the shock wave are least precise—an undesirable consequence of using fixed-rate compression.

By fixing the precision and spatially adapting the rate, we allocate more bits to where they are needed. We are developing such variable-rate arrays to achieve **adaptive-rate computing**, with many-fold reductions in data movement. Our approach is analogous but complementary to adaptive mesh refinement, which varies cell size rather than precision to resolve physically important processes.



Adaptive-rate arrays allocate more bits to difficult-to-compress parts of the domain to ensure precision (here 40 bits) or accuracy needs are met, still reducing data storage and movement.