Reducing Checkpoint Size in PlasComCM with Lossy Compression

14th Annual Workshop on Charm++ and its Applications

Jon Calhoun[1], Franck Cappello[1,2], Luke Olson[1], and Marc Snir[1,2], Sheng Di[2]

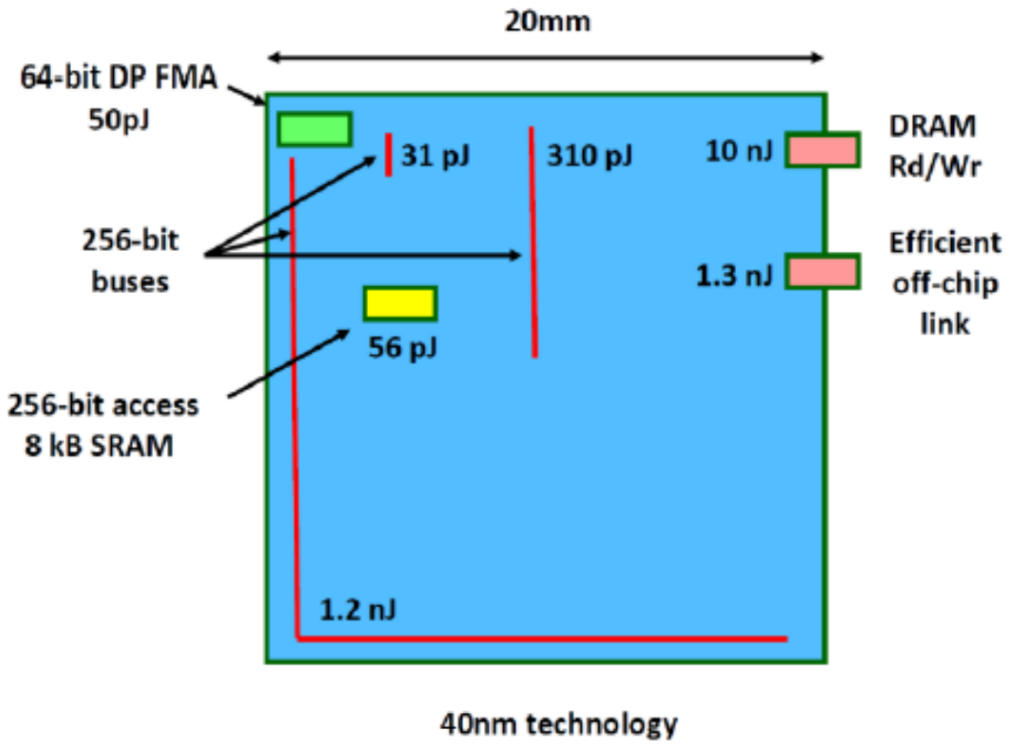
[1]University of Illinois at Urbana-Champaign
[2]Argonne National Labatory

19 April 2016

Data Movement Problem

On current systems, computation is essentially free compared to time and energy required for data transfers.

What do we do with these *free* CPU cycles?

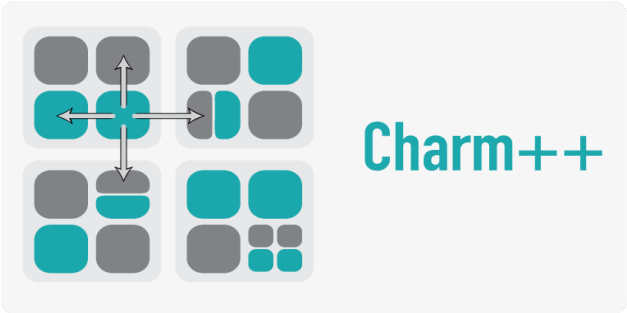


[Keckler 2011]

Checkpoint Restart in Charm++

Native checkpoint restart

- partner nodes
- permanent storage



Although checkpointing to a partner node is much faster, checkpointing to permanent storage is still needed.

Let's look at improving checkpointing to the parallel filesystem.

Lossless compression?

Table 1. Comparison of lossless compression schemes.

Scheme	Transformation Applied	Algorithm	Compression Ratio
FPC [8]	not used	it first predicts values sequentially using two predictors (FCM and DFCM), and subsequently selects the closer predicted value to the actual. Lastly, it XORs the selected predicted value with the actual value, and leading-zero compresses the result.	1.02x∼1.96x
ISOBAR [30]	divide byte-columns into compressible and incompressibles	apply zlib, bzlib2, (fpzip, FPC) on all compressible (after discarding noisy byte-columns). zlib is the main compression algorithm; others are for comparison purposes	1.12x∼1.48x
PRIMACY [31]	frequency based permutation of ID values	apply zlib on transformed data	1.13x∼2.16x
ALACRITY [19]	split floating-point values into sign, exponent, and significands	unique-value encoding of the most significant bytes (assuming high-order bytes (sign and exponents) are easy to compress); low-order bytes are compressed using ISOBAR	1.19x∼1.58x
CC [6]	XOR on Δ of neighboring data point in the same iteration	apply zero-filled run length encoding	up to 2.13x
IOFSL [36]	not used	integration of LZO, bzip2, zlib within the I/O forwarding layer	∼1.9x
Binary Masking [5]	bit masking (XOR)	apply zlib on bit masked data in order to partially decreases the entropy level	1.11x∼1.33x
MCRENGINE [18]	variable merging in the same group	apply parallel gzip on the merged variables across processes	up to 1.18x

[Son et al. 2014]

- Standard compression schemes not designed for floating-point
- Lossless floating-point schemes provide small compression factors

Table 1. Comparison of lossless compression schemes.

Scheme	Transformation Applied	Algorithm	Compression Ratio
FPC [8]	not used	it first predicts values sequentially using two predictors (FCM and DFCM), and subsequently selects the closer predicted value to the actual. Lastly, it XORs the selected predicted value with the actual value, and leading-zero compresses the result.	1.02x∼1.96x
ISOBAR [30]	divide byte-columns into compressible and incompressibles	apply zlib, bzlib2, (fpzip, FPC) on all compressible (after discarding noisy byte-columns). zlib is the main compression algorithm; others are for comparison purposes	1.12x∼1.48x
PRIMACY [31]	frequency based permutation of ID values	apply zlib on transformed data	1.13x∼2.16x
ALACRITY [19]	split floating-point values into sign, exponent, and significands	unique-value encoding of the most significant bytes (assuming high-order bytes (sign and exponents) are easy to compress); low-order bytes are compressed using ISOBAR	1.19x∼1.58x
CC [6]	XOR on Δ of neighboring data point in the same iteration	apply zero-filled run length encoding	up to 2.13x
IOFSL [36]	not used	integration of LZO, bzip2, zlib within the I/O forwarding layer	∼1.9x
Binary Masking [5]	bit masking (XOR)	apply zlib on bit masked data in order to partially decreases the entropy level	1.11x∼1.33x
MCRENGINE [18]	variable merging in the same group	apply parallel gzip on the merged variables across processes	up to 1.18x

[Son et al. 2014]

Lossy Compression

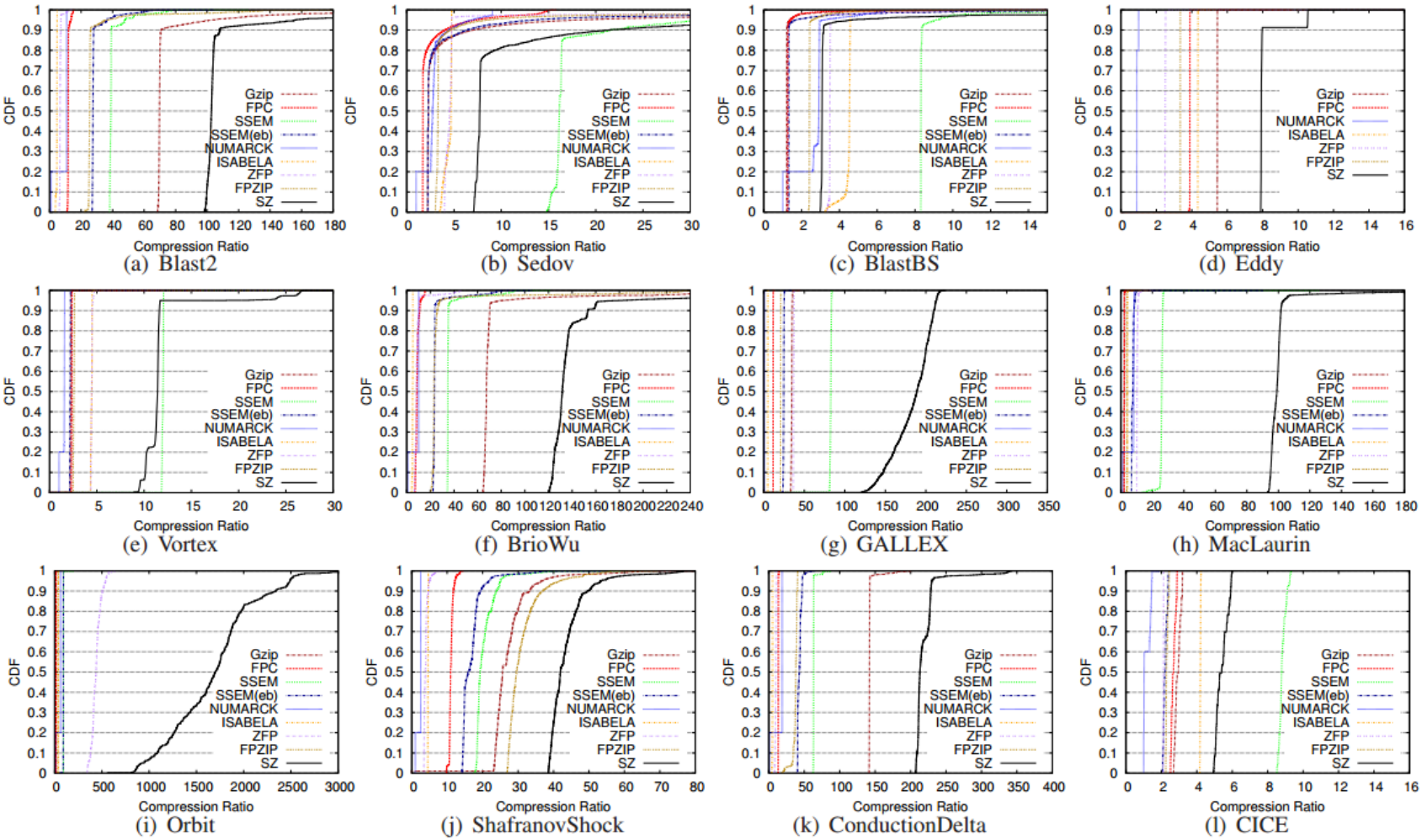


Figure 4. CDF of Compression Ratios (note that SSEM, NUMARCK and ISABELA do not respect specified error bound as shown in Figure 5)

High compression ratios with lossy compression [Di and Cappello 2016]

Can applications be restarted from a lossy checkpoint?

Whenever you use floating-point values you have already embraced various amounts of error

- Floating-point arithmic alread suffers from error due to roundoff.
- Numerical methods used to solve PDEs and ODEs are only accurate to the order of the method.

Understanding Error

Many lossy compression schemes allow you to specify an error bound (e.g. relative, absolute).

- How should I evaluate this error?
- Is this error detrimental?

Evaluation

Accuracy of numerical methods is expressed as $\mathcal{O}(h^p)$.

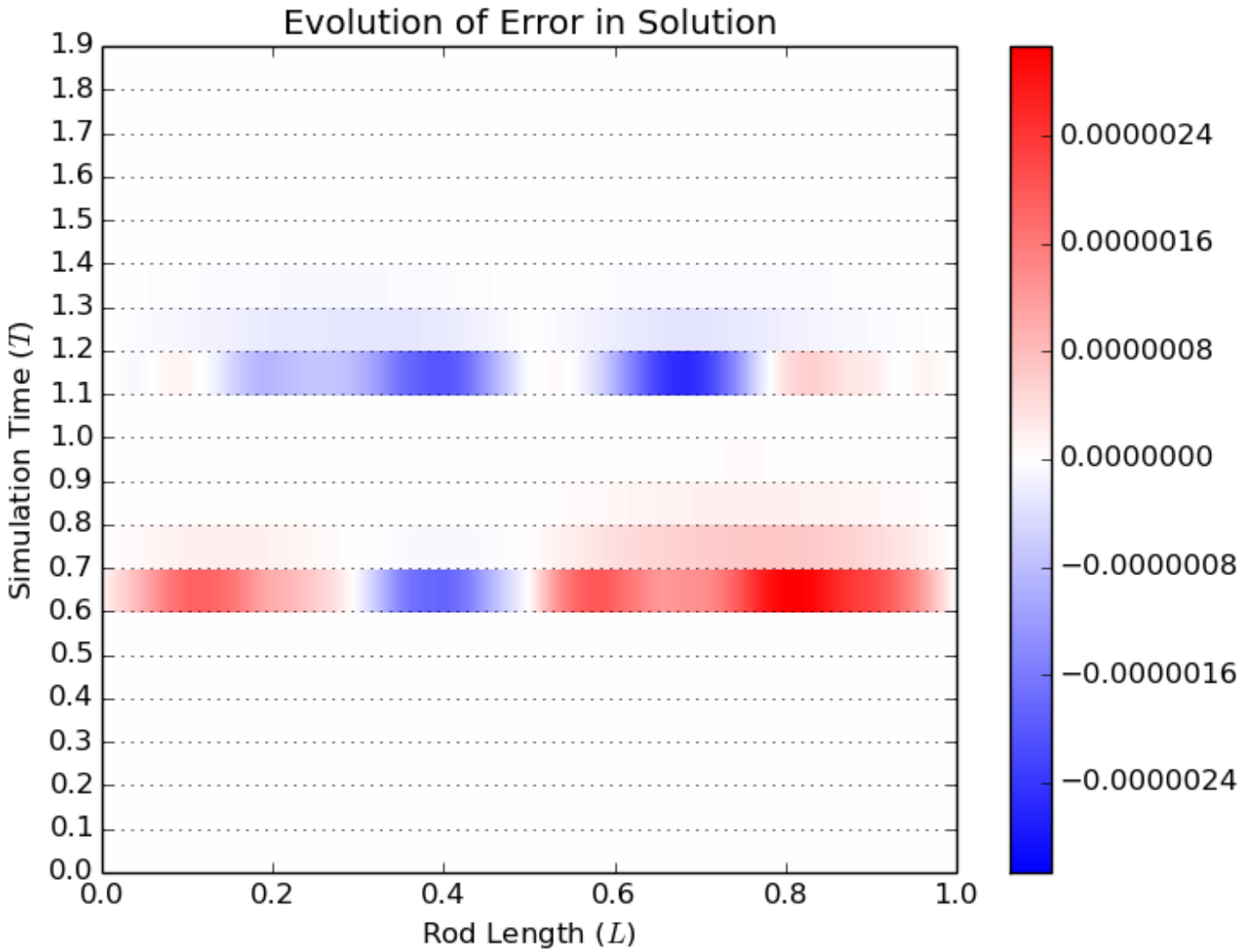
Restrict lossy compression error tolerance to be less than truncation error, then error added by lossy compression is *hidden* in the simulation.

Let's first look at a 1-D heat and a 1-D advection equation to understand what happend to simple PDEs.

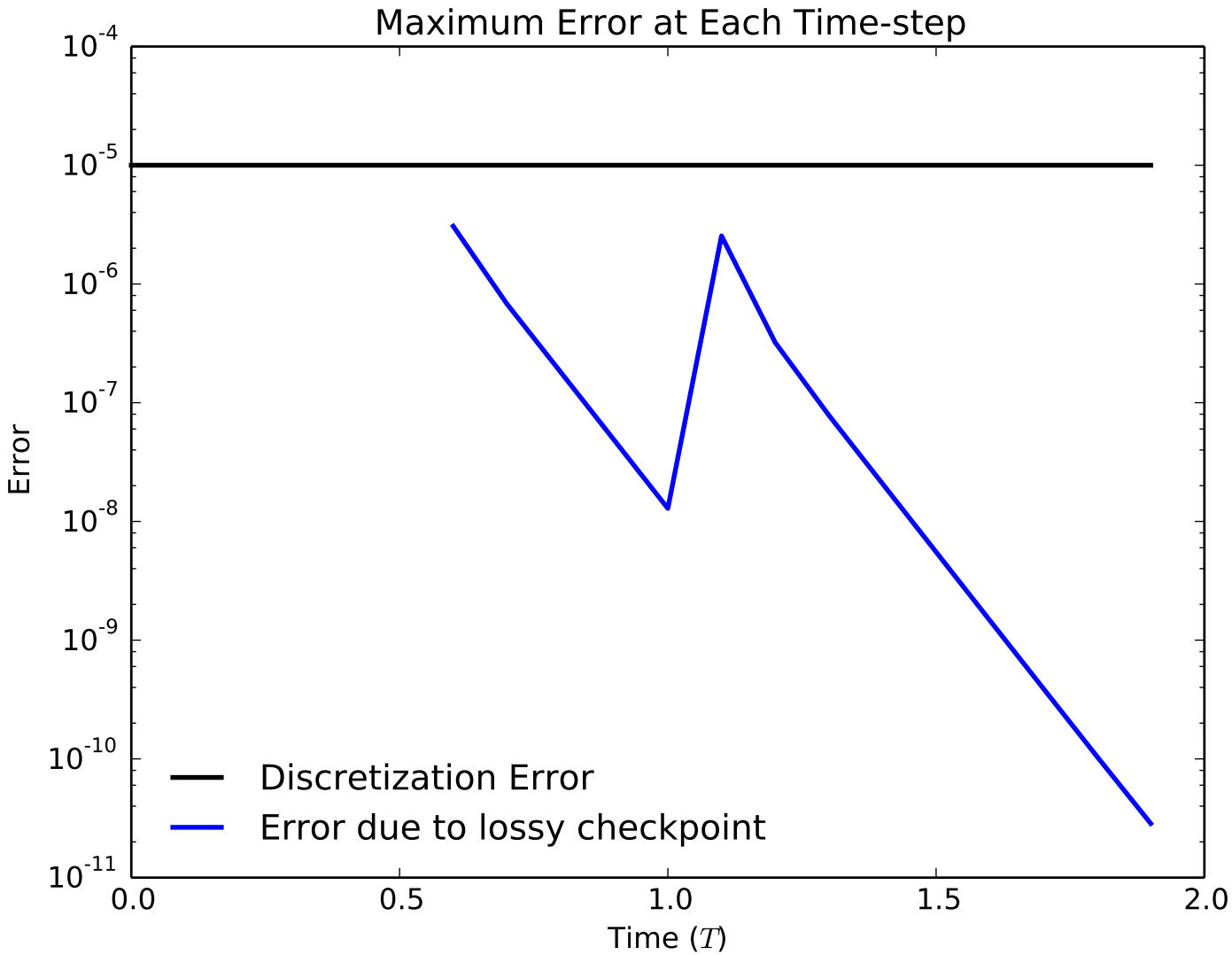
Setup:

- Lossy Compressor: SZ-0.5.5 [Di and Cappello 2016]
- Data vectors 64-bit floating-point
- Checkpoint PDE state variables

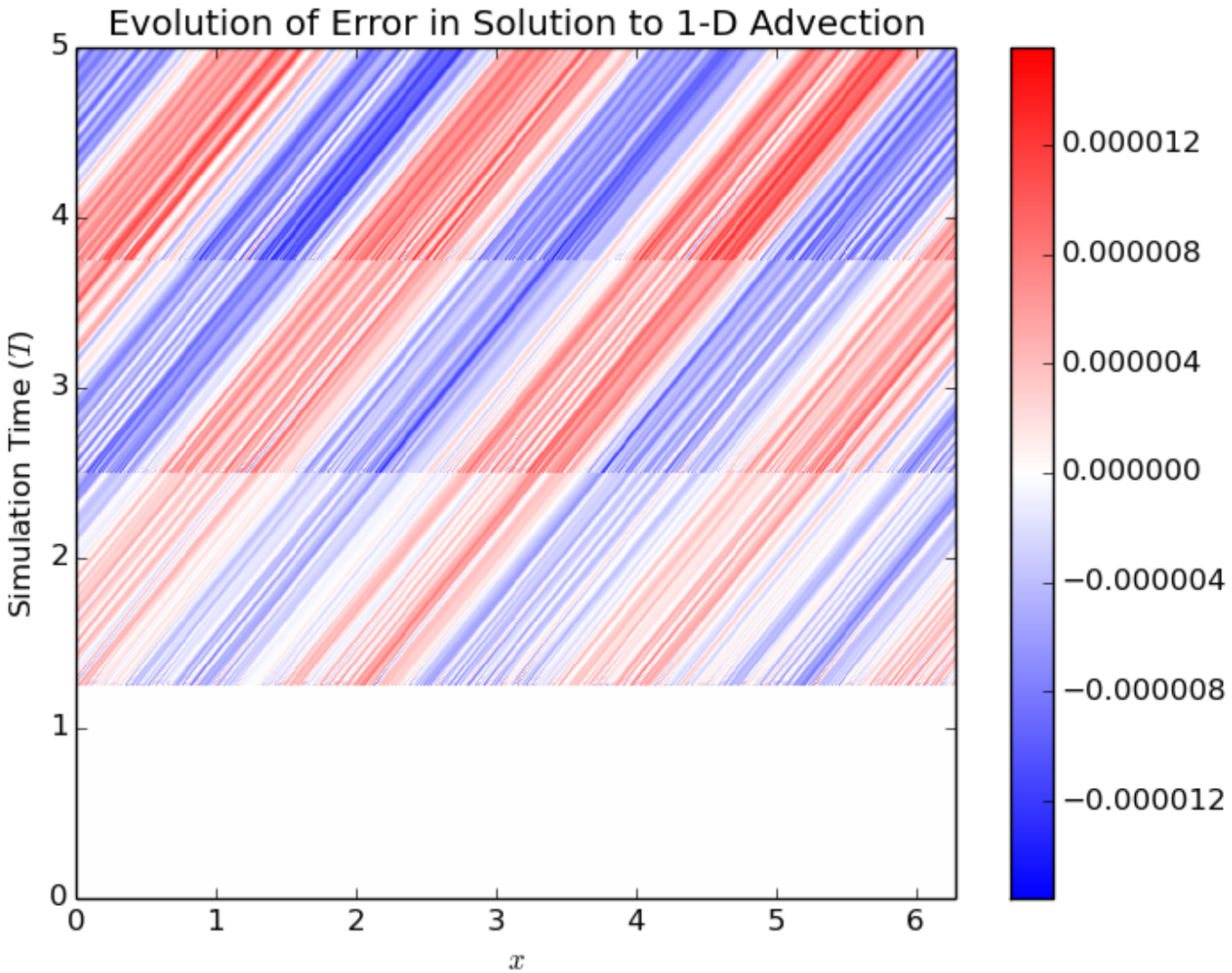
1-D Heat Equation Error Evolution



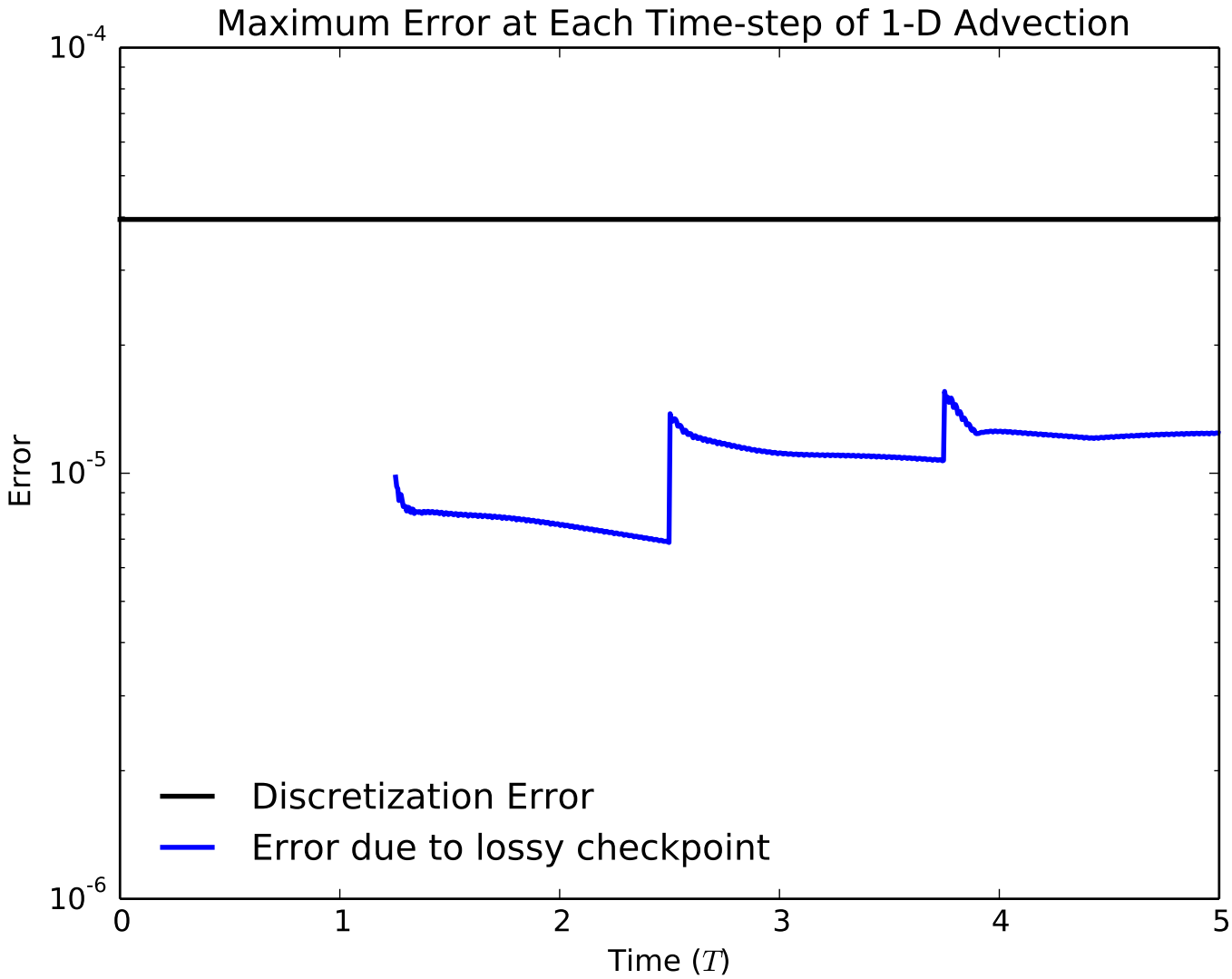
1-D Heat Error Evolution



1-D Advection Equation Error Evolution



1-D Advection Error Evolution



XPACC PlasComCM

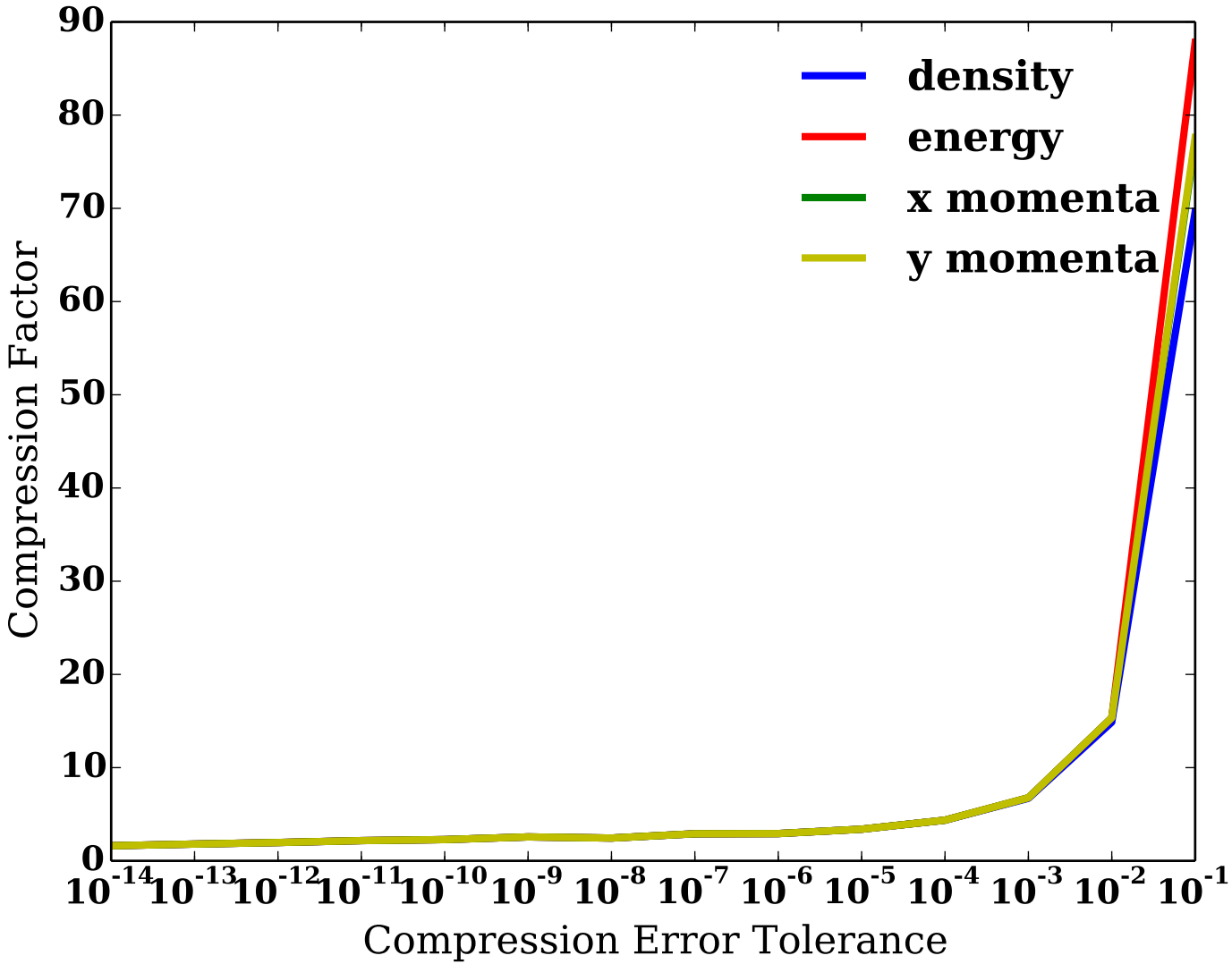
PlasComCM

- coupled multipysics code
- Checkpoint restart accomplished by AMPI

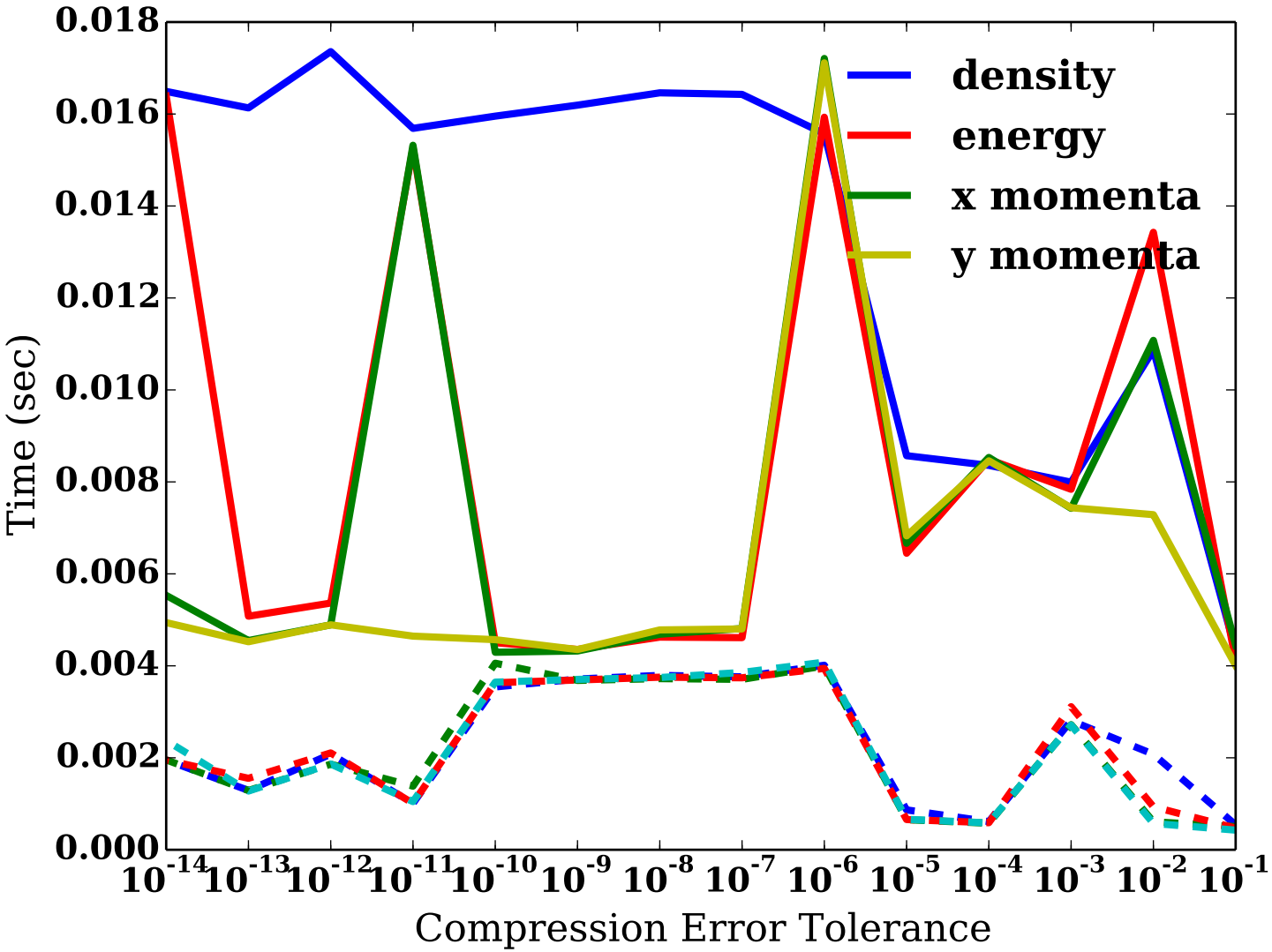
Setup:

- Navier-stokes flow past cylinder problem
- $h_x = h_y = 0.0015$
- Checkpoint every 5000 iterations to $1e^{-}14$

PlasComCM Compression Factor



PlasComCM Timings

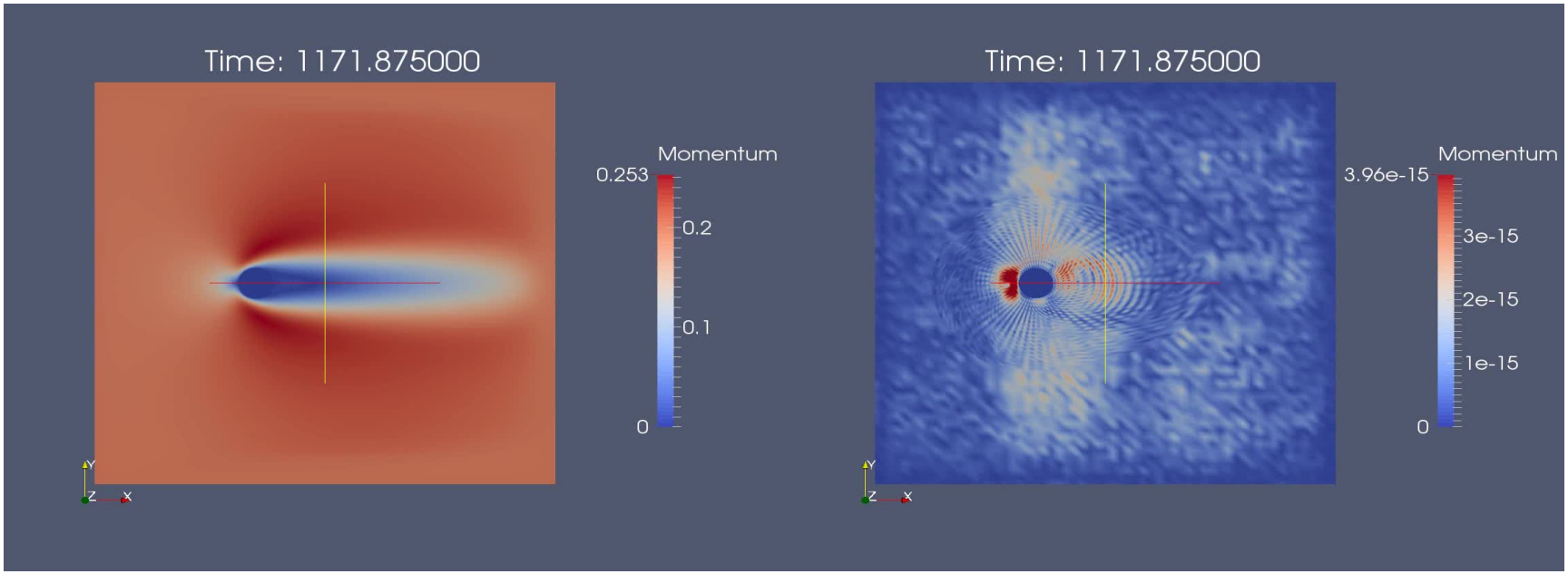


Solid line compression time. Dotted line decompression time.

Simulation

Simulation

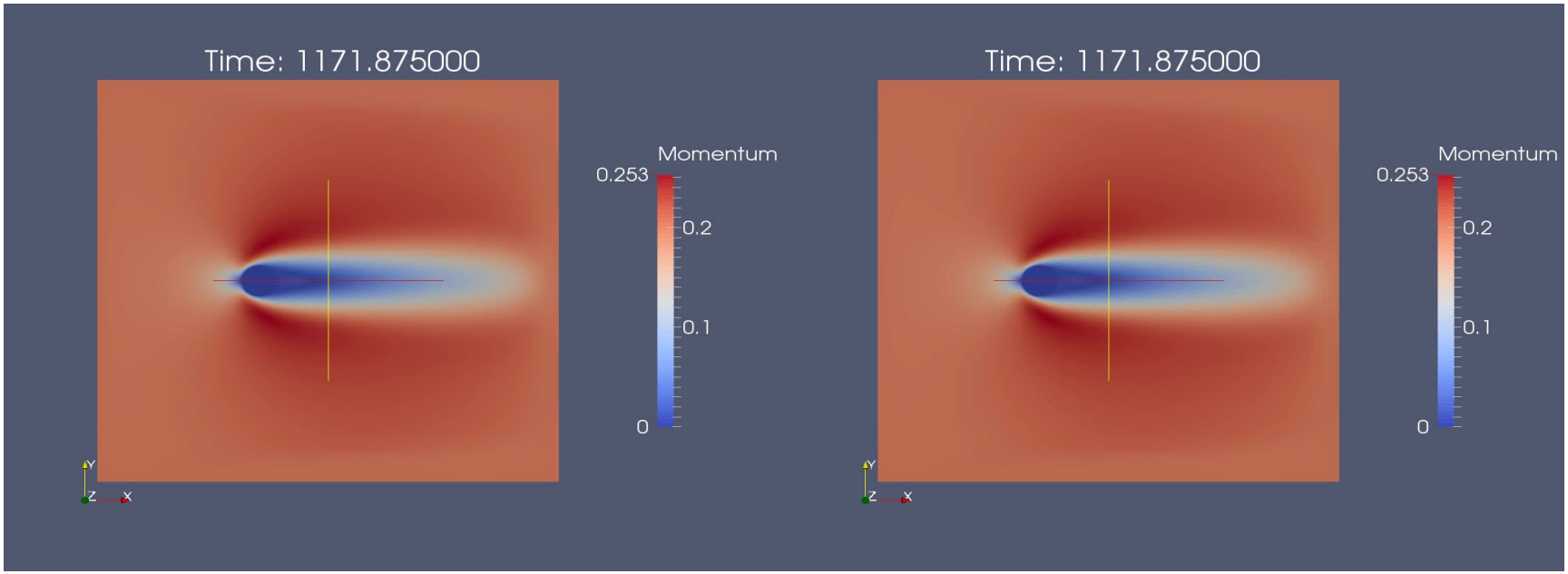
Error



What is the compression error tolerance $1e^{-}$?

Simulation

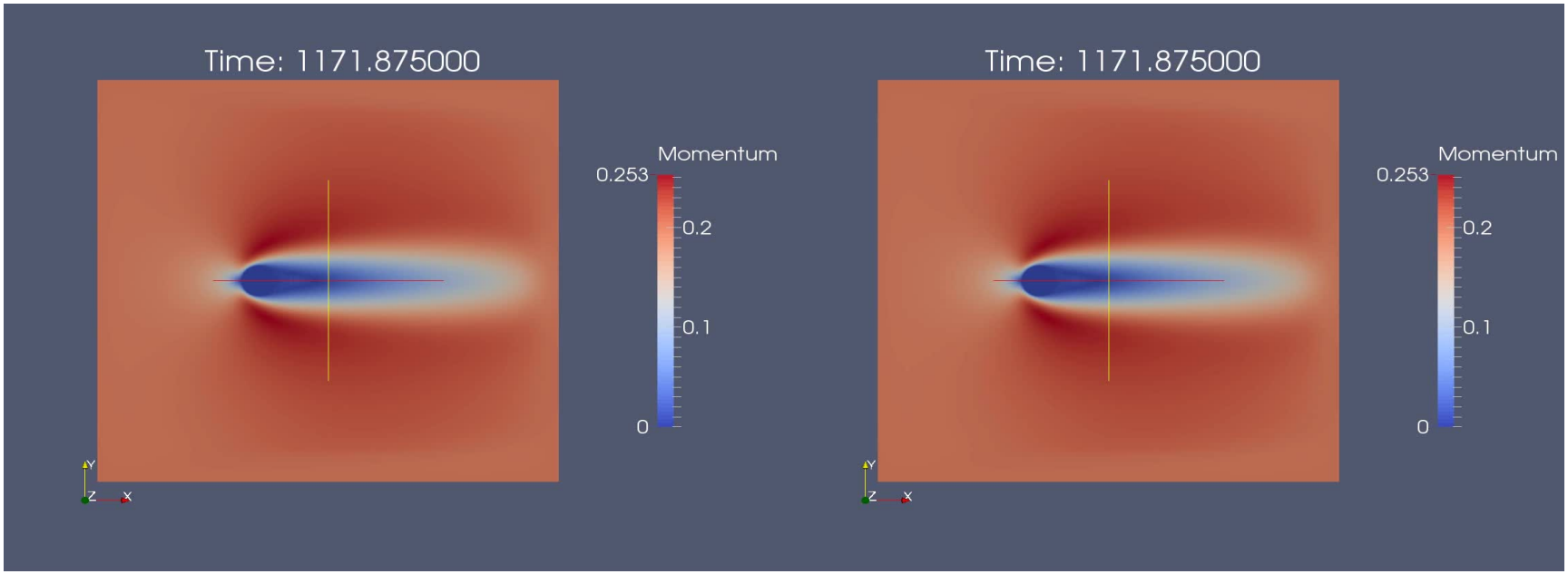
Lossy Compressed Simulation



What is the compression error tolerance $1e^{-}2$

Simulation

Lossy Compressed Simulation



Conclusion and Future Work

Lossy compression can effectively reduce the size of a checkpoint without affecting the negatively solution

Currently only applicable to file system checkpoints

Need to discuss with users to determine acceptable error tolerance

Investigate other applications and inputs to gain further insight

Further leverage application properties when compressing

Acknowledgments

- This work was sponsored by the Air Force Office of Scientific Research under grant FA9550-12-1-0478.

Thank you

Any questions?