Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data

Sriram Lakshminarasimhan[1,2], Neil Shah[1], Stephane Ethier[3], Scott Klasky[2], Rob Latham[4], Rob Ross[4], and Nagiza F. Samatova[1,2,⋆]

[1] North Carolina State University, Raleigh, NC 27695, USA
[2] Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA
[3] Princeton Plasma Physics Laboratory, Princeton, NJ 08543, USA
[4] Argonne National Laboratory, Argonne, IL 60439, USA
samatova@csc.ncsu.edu

Abstract. Modern large-scale scientific simulations running on HPC systems generate data in the order of terabytes during a single run. To lessen the I/O load during a simulation run, scientists are forced to capture data infrequently, thereby making data collection an inherently *lossy* process. Yet, *lossless* compression techniques are hardly suitable for scientific data due to its inherently random nature; for the applications used here, they offer less than 10% compression rate. They also impose significant overhead during decompression, making them unsuitable for data analysis and visualization that require repeated data access.

To address this problem, we propose an effective method for In-situ Sort-And-B-spline Error-bounded Lossy Abatement (**ISABELA**) of scientific data that is widely regarded as effectively incompressible. With ISABELA, we apply *a preconditioner* to seemingly random and noisy data along spatial resolution to achieve an accurate fitting model that guarantees a ≥ 0.99 correlation with the original data. We further take advantage of temporal patterns in scientific data to compress data by $\approx 85\%$, while introducing only a negligible overhead on simulations in terms of runtime. ISABELA significantly outperforms existing lossy compression methods, such as Wavelet compression. Moreover, besides being a communication-free and scalable compression technique, ISABELA is an inherently local decompression method, namely it does not decode the entire data, making it attractive for random access.

Keywords: Lossy Compression, B-spline, In-situ Processing, Data-intensive Application, High Performance Computing.

1 Introduction

Spatio-temporal data produced by large-scale scientific simulations easily reaches terabytes per run. Such data volume poses an I/O bottleneck—both while writing the data into the storage system during simulation and while reading the data back during analysis and visualization. To alleviate this bottleneck, scientists have to resort to subsampling, such as capturing the data every s^{th} timestep.

⋆ Corresponding author.

E. Jeannot, R. Namyst, and J. Roman (Eds.): Euro-Par 2011, LNCS 6852, Part I, pp. 366–379, 2011.
© Springer-Verlag Berlin Heidelberg 2011

This process leads to an inherently *lossy* data reduction.

In-situ data processing—or processing the data in-tandem with the simulation by utilizing either the same compute nodes or the staging nodes—is emerging as a promising approach to address the I/O bottleneck [12]. To complement existing approaches, we propose an effective method for In-situ Sort-And-B-spline Error-bounded Lossy Abatement (ISABELA) of scientific data.

ISABELA is particularly designed for compressing spatio-temporal scientific data that is characterized as being inherently noisy and random-like, and thus commonly believed to be uncompressible [16]. In fact, any *lossless* compression technique [3,13] is capable of reducing such data by no more than a 10% of its original size, besides being computationally intensive and, therefore, hardly suitable for *in-situ* processing (see Section 3).

The intuition behind ISABELA stems from the following three observations. First, while being almost random and noisy in its natural form—when sorted—scientific data exhibits a very strong signal-to-noise ratio due to its monotonic and smooth behavior in its sorted form. Second, prior work done in curve fitting [7,17] have shown that monotone curve fitting, such as monotone B-splines, can offer some attractive features for data reduction, including, but not limited to, their goodness of fit with significantly fewer coefficients to store. Finally, the monotonicity property of the sorted data gets preserved in most of its positions with respect to adjacent time steps. Hence, this property of monotonic inheritance across temporal resolution offers yet another venue for improvement of the overall data compression ratio.

While intuitively simple, ISABELA has addressed a number of technical challenges imposed by end-user's requirements. One of the most important factors for the user's adoption of any lossy data reduction technology is the assurance that the user-acceptable error-bounds are respected. Since curve fitting accuracy is often data-dependent, ISABELA must be robust in its approximation. While curve fitting operations are traditionally time consuming, performing the compression *in-situ*, mandates ISABELA to be fast. Finally, while data sorting—as a pre-conditioner for data reduction—is "a blessing," it is "a curse" at the same time; reordering the data requires keeping track of the new position indices to associate the decoded data with its original ordering. While management of spline coefficients could be viewed as a light-weight task, the heavy-weight index management forces ISABELA to make some non-trivial decisions between the data compression rates and the data accuracy.

2 A Motivating Example

Much of the work for *in-situ* data reduction in this paper stems from a Gyrokinetic Tokamak Simulation (GTS) [15] for studying plasma micro-turbulence in the core of magnetically confined fusion plasmas of toroidal devices in nuclear reactors. On current petaflop systems, such as NCCS/ORNL Jaguarpf, the GTS code, utilizing ADIOS [11] for its intensive I/O, has demonstrated weak scaling for up to 65,536 cores on the 8-core per node configuration.

The entire GTS data set can be broadly divided into: (1) checkpoint data to restart the simulation in case of an execution failure (C&R); (2) analysis (A) data, such as density and potential fluctuations, for performing various post-processing physics analyses, and (3) diagnostics data used, for example, for code validation and verification (V&V) (see Table 1).

Table 1. Summary of GTS output data by different categories

Category	Write Frequency	Read Access	Size/Write	Total Size
C&R	Every 1-2 hours	Once or never	A few TBs	≈TBs
A	Every 10^{th} time step	Many times	A few GBs	≈TBs
V&V	Every 2^{nd} time step	A few times	A few MBs	≈GBs

Unlike C&R data that requires lossless compression, analysis (A) data is inherently lossy, and as such, it can tolerate some error-bounded loss in its accuracy. What is more important is that it is the analysis data that is being accessed many times by different scientists using various analysis and visualization tools or Matlab physics analysis codes. Therefore, aggressive data compression that could enable interactive analytical data exploration is of paramount concern, and is, therefore, the main focus of ISABELA. For illustrative purposes, throughout the paper, we will use temporal snapshots of the GTS analysis data consisting of one-dimensional 64-bit double precision floating point arrays of 172,111 values each for *Potential* and *Density* fluctuations.

3 Problem Statement

The inherent complexity of scientific spatio-temporal data drastically limits the applicability of both lossless and lossy data compression techniques and presents a number of challenges for new method development. Such data not only consists of floating-point values, but also exhibits randomness without any distinct repetitive bit and/or byte patterns (also known as high entropy data, and hence, uncompressible [5,14]). Thus, applying standard *lossless* compression methods does not result in an appreciable data reduction.

Table 2 illustrates the compression rates achieved and the time required to compress and decode 12,836KB of GTS analysis data by state-of-the-art methods. In addition, scientific data often exhibits a large degree of fluctuations in values across even directly adjacent locations in the array. These fluctuations render *lossy* multi-resolution compression approaches like Wavelets [6] ineffective.

The compression ratio $CR_M(D)$ of a compression method M for data D of size $|D|$ reduced to size $|D_M|$ is defined by Eq. 1:

$$CR_M(D) = \frac{|D| - |D_M|}{|D|} \times 100\%. \tag{1}$$

The accuracy of lossy encoding techniques is measured using Pearson's correlation coefficient (ρ) and Normalized Root Mean Square Error between an

$N-$dimensional original data vector $D = (d_0, d_1, \ldots, d_{N-1})$ and decompressed data vector $D' = (d'_0, d'_1, \ldots, d'_{N-1})$ defined by Eq. 2:

$$NRMSE_M(D) = \frac{RMSE_M(D, D')}{Range(D)} = \frac{\sqrt{\sum_{i=0}^{N-1} (d_i - d'_i)^2}}{max(D) - min(D)}. \quad (2)$$

Table 2. Performance of examplar lossless and lossy data compression methods

Metric	**FPC**	**LZMA**	**ZIP**	**BZ2**	**ISABELA**	**Wavelets**	**B–splines**
Lossless?	Yes	Yes	Yes	Yes	No	No	No
CR_M (%)	3.12	2.72	1.13	1.11	**81.44***	22.51*	0*
Compression (sec.)	0.58	7.01	1.03	3.96	0.93	0.62	0.78
Decompression (sec.)	0.56	1.38	0.49	1.18	1.05	0.58	0.82

*CR achieved by lossy models for 0.99 correlation and 0.01 NRMSE fixed accuracy. All runs are performed on an Intel Core 2 Duo 2.2 GHz processor with 4 GB RAM, openSUSE Linux v11.3.

4 Theory and Methodology

Existing multi-resolution compression methods often work well on image data or time-varying signal data. For scientific data-intensive simulations, however, data compression across the *temporal resolution* requires data for many timesteps be buffered in memory that is, obviously, not a viable option. Applying lossy compression techniques on this data across the *spatial resolution* requires a significant tradeoff between the *compression ratio* and the *accuracy*. Hence, to extract the best results out of the existing approximation techniques, a transformation of this data layout becomes necessary.

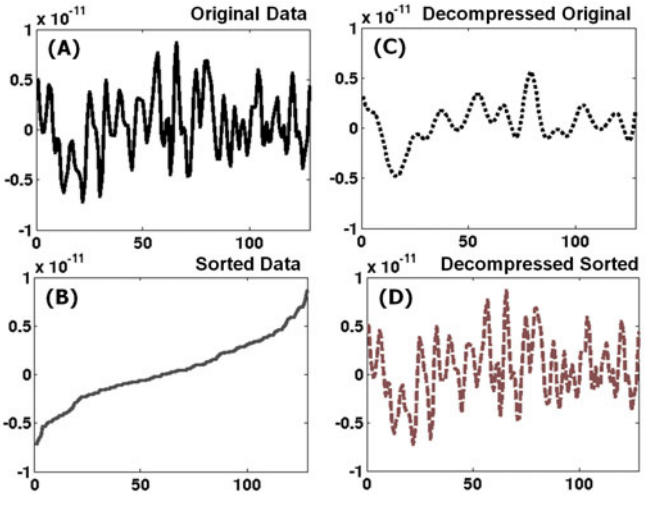

Fig. 1. A slice of GTS Potential: (A) original; (B) sorted; (C) decoded after B–splines fitting to original; and (D) decoded after B–splines fitting to sorted

4.1 Sorting-Based Data Transformation

Sorting changes the data distribution in the spatial domain, from a highly irregular signal (Fig. 1, **A**) to a smooth and monotonous curve (Fig. 1, **B**). The rationale behind sorting—as a pre-conditioner for a compression method—is that fitting on a monotonic curve can provide a model that is more accurate than the one on unordered and randomly distributed data. Figure 1 illustrates the significant contrast in how closely (**D**) or poorly (**C**) the decompressed data approximates the original data when the $B-$splines curve fitting [2] operates on sorted versus unsorted data, respectively.

4.2 Cubic $B-$Splines Fitting

Sorting the data in an increasing order provides a sequence of values whose rate of change is guaranteed to be the slowest. Although this sequence resembles a smooth curve, performing curve fitting using non-linear polynomial interpolation becomes difficult for complex shape curves. Computing interpolation constants for higher-order polynomials in order to fit these complex curves is computationally intensive for *in-situ* processing.

A more effective technique is by using $B-$splines curve fitting. A $B-$splines curve is a sequence of piecewise lower order parametric curves joined together via knots. Cubic $B-$splines are composed of polynomial functions of degree $d = 3$, which have faster interpolation time and produce "smooth" curves (i.e., second-order differentiable) at the knot locations. The shape of the $B-$splines curve is determined by a knot sequence that describes the span of the piecewise segments, and a set of basis functions that influences the segments of the curve. Because of this property splines can control the local shape of the curve without affecting the shape of the curve globally. This also implies that both curve fitting and interpolation are efficient operations and can provide location-specific data decoding without decompressing all the data. While sorting rearranges the points, location-specific decoding is still possible by performing an additional single level translation of data location from the original to the sorted vector, and then retrieving the interpolated value on the sorted $B-$spline curve.

4.3 Maximizing Compression Ratio via Window Splitting

In this section, we look at approaches to maximizing the compression ratio, while maintaining an accurate approximation model. Let us assume that the original data D is a vector of size N, namely $D = (d_0, d_1, \ldots, d_{N-1})$. This way we can associate a value d_i with each index value $i \in I = \{0, 1, \ldots, N-1\}$. Let us also assume that each vector element, $d_i \in \mathbb{R}$, is stored as a 64-bit double-precision value. Therefore, storing the original data requires $|D| = N \times 64$ bits.

Assuming that D is a discrete approximation of some curve, its $B-$splines interpolation D_B requires storing only $B-$splines constants—the knot vector and the basis coefficients—in order to reconstruct the curve. Let C denote the

number of such 64-bit double-precision constants. Then storing the compressed data after B-splines curve fitting requires $|D_B| = C \times 64$ bits.

The random-like nature of D (see Fig. 1, (A)) requires $C \sim N$ to provide accurate lossy compression, and hence, leads to a poor compression rate (see Table 2, last column). However, applying B-splines interpolation after sorting D requires only a few constants, $C = O(1) << N$, in order to provide high decompression accuracy (see Fig. 1, (D)).

While significantly reducing the number of $B-$splines constants C, sorting D will reorder the vector elements via some permutation π of its indices, namely $I \xrightarrow{\pi} I_\pi = \{i_1, i_2, \ldots, i_N\}$, such that $d_{i_j} \leq d_{i_{j+1}}, \forall i_j \in I_\pi$. As a result, we need to keep track of the new index I_π so that we could associate the decompressed sorted vector D_π back to the original vector D by using its correct index I. Since each index value i_j requires $log_2 N$ bits, the total storage requirement for I_π is thus $|I_\pi| = N \times log_2 N$ bits. Therefore, the vector length N is the only factor that determines the storage requirements for the index I_π.

One way to optimize the overall compression ratio, CR_{ISABELA}, is to first split the entire vector D into fixed-sized windows of size W_0, or $D = \bigcup D^k$, $D^i \cap D^j = \emptyset$, $I^k = \{(k-1)W_0, (k-1)W_0 + 1, \ldots, kW_0\}$, $i, j, k \in \overline{1, N_W}$, $i \neq j$, and $N_W = \left\lceil \frac{N}{W_0} \right\rceil$. Then, the $B-$splines interpolation is applied to each window D^k separately.

With this strategy, ISABELA's storage requirement for the compressed data is defined by Eq. 3:

$$\begin{aligned} |D_{\text{ISABELA}}| &= \sum_{k=1}^{N_W} (|D_B^k| + |I_\pi^k|), \\ &= N_W \times (C \times 64 + W_0 \times log_2 W_0) \end{aligned} \tag{3}$$

Substituting Eq. 3 into Eq. 1 and simplifying the resulting equation, we obtain the following compression ratio for ISABELA defined by Eq. 4:

$$CR_{\text{ISABELA}}(D) = (1 - \frac{log_2(W_0)}{64} - \frac{C}{W_0}) \times 100\% \tag{4}$$

From Eq. 4, we can analytically deduce the trade-off between the window size W_0 and the number of $B-$splines constants C that give the best compression ratio. For example, for $W_0 > 65,536$, the size of the index alone would consume more than 25% of the original data. We found that $C = 30$ and $W_0 = 1024$ allows ISABELA to achieve both > 0.99 correlation and < 0.05 NRMSE between the original and decompressed GTS data. Also, fixing $W_0 = 1024$ balances the cost of storing both the index and the fitting coefficients giving an overall compression rate of 81.4% per time step.

4.4 Error Quantization for Guaranteed Point-by-Point Accuracy

The above sorting-based curve fitting model ensures accurate approximation only on a *per window* basis and not on a *per point* basis. As a result, in certain

locations, the $B-$splines estimated data deviates from the actual by a margin exceeding a defined tolerance. For example, almost 95% of the approximated GTS Potential values average a 2% relative error, where the percentage of the relative error (ϵ) at each index i between $D = (d_0, d_1, ..., d_{N-1})$ and $D_{ISABELA} = (d'_0, d'_1, ..., d'_{N-1})$ is defined as $\epsilon_i = \frac{d_i - d'_i}{d_i} \times 100\%$. While the number of such location points is reasonably low due to accurate fitting achieved by $B-$splines on monotonic data, ISABELA guarantees that a user-specified point-by-point error is respected by utilizing an *error quantization* strategy.

Storing relative errors between estimated and actual values enables us to reconstruct the data with high accuracy. Quantization of these errors into 32-bit integers results in a large degree of repetition, where majority of the values lie between [-2, 2]. These integer values lend themselves to high compression rates ($75\% - 90\%$) with standard lossless compression libraries. These compressed relative errors are stored along with the index during encoding. Upon decoding, applying these relative errors ensures decompressed values to be within a user-defined threshold τ_ϵ for per point relative error.

4.5 Exploiting Δ–Encoding for Temporal Index Compression

To a large extent, the ordering of the sorted data values is similar between adjacent timesteps, i.e., the monotonicity property of the data extends to index integer values. Hence, we apply a differential encoding scheme to the index vector I_π before compressing the index using standard lossless compression libraries. [Note that subsequent scheme is applied to each individual data window D^n.]

Suppose that at timestep t_0, we first build the index $I_\pi(t_0)$ consisting of no redundant values, essentially, incompressible. Hence, this index is stored as is. But, at the next timestep $t_0 + 1$, the difference in index values $\Delta I_{+1} = I_\pi(t_0 + 1) - I_\pi(t_0)$ is small (see Fig. 2) due to monotonicity of the original data values D^n and, hence, the sorted values across adjacent timesteps.

Thus, instead of storing the index values at each timestep, we store the index values at t_0, denoted as the *reference* index, along with the compressed pairwise index differences ΔI_{+1} between adjacent timesteps. But, in order to recover the

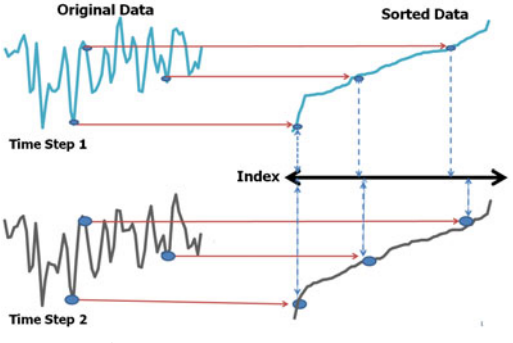

Fig. 2. Illustration of Δ–encoding of the index across temporal resolution

data at time $t_0 + \delta t$, we must read in both the reference index $I_\pi(t_0)$ and all the *first-order* differences ΔI_{+1} between adjacent timesteps in the time window $(t_0, t_0+\delta t)$. Therefore, the higher value of δt will adversely affect reading time. To address this problem, we instead store and compress a *higher-order* difference, $\Delta I_{+j} = I_\pi(t_0+j) - I_\pi(t_0)$, where $j \in (1, \delta t)$, for the growing value of δt until the size of the compressed index crosses a user-defined threshold. Once the threshold is crossed, the index for the current timestep is stored as is, and is considered as the new reference index.

5 Results

Evaluation of a lossy compression algorithm primarily depends on the accuracy of the fitting model and the compression ratio (CR) achieved. As this compression is performed *in-situ*, analysis of the time taken to perform the compression assumes significance as well. Here, we evaluate ISABELA with emphasis on the aforementioned factors, using normalized root mean standard error ($NRMSE$) and Pearson correlation (ρ) between the original and decompressed data as accuracy metrics. [Note that achieving $NRMSE \sim 0$, $\rho \sim 1$, and $CR \sim 100\%$ would indicate excellent performance.]

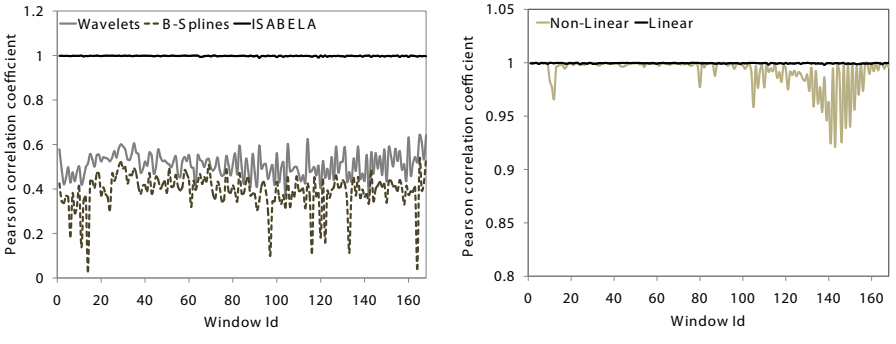

Fig. 3. Accuracy (ρ): (a) Per window correlation for Wavelets, $B-$splines, and ISABELA with fixed $CR = 81\%$ for GTS Density. (b) Per window correlation for GTS linear and non-linear stage Potential decompressed by ISABELA.

5.1 Per Window Accuracy

In this section, we compare the Pearson correlation (ρ) between the original and decompressed data using Wavelets and B-Splines on original data and using ISABELA. The following parameters are fixed in this experiment: $CR = 81\%$ for Density, $W_0 = 1024$, $C_{B-spline} = 150$, and $C_{\text{ISABELA}} = 30$. Wavelet coefficients are thresholded to achieve the same compression rate. Figure 3(a) illustrates that ISABELA performs exceptionally well even for much smaller C values due to

the monotonic nature of the sorted data. In fact, $\rho > 0.99$ for almost all the windows. However, both Wavelets and $B-$splines exhibit a large degree of variation and poor accuracy across different windows. This translates to NRMSE values that are one-to-two orders of magnitude larger than the average 0.005 NRMSE value produced by ISABELA.

ISABELA performs exceptionally well on data from the linear stages of the GTS simulation (first few thousand timesteps), as shown in Fig. 3(b). Yet, the performance for the non-linear stages (timestep $\approx 10,000$), where the simulation is characterised by a large degree of turbulence, is of particular importance to scientists. Figure 3(b), with intentionally magnified correlation values, shows that accuracy for the non-linear stages across windows drops indeed. Unlike Wavelets (Fig. 3(a)), however, this correlation does not drop below 0.92.

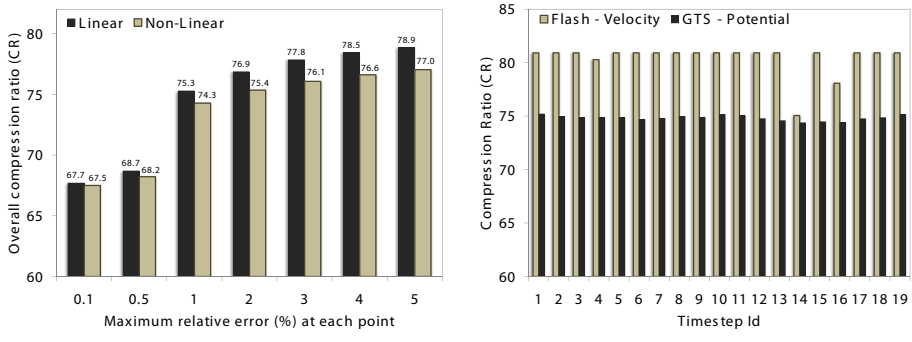

Fig. 4. Compression ratio (CR) performance: (a) For various per point relative error thresholds (τ_ϵ) in GTS Potential during linear and non-linear stages of the simulation. (b) For various timesteps with $\tau_\epsilon = 1\%$ at each point (for GTS Potential: $t_1 = 1,000$, $\Delta t = 1,500$; for Velocity in Flash: $t_1 = 3,000$, $\Delta t = 3,500$.

5.2 Trade-Off between Compression and Per Point Accuracy

To alleviate the aforementioned problem, we apply error quantization, as described in Sec. 4.4. In both linear and non-linear stages of GTS simulation, the compression ratios are similar when the per point relative error (τ_ϵ) is fixed (see Fig. 4(a)). This is because the relative error in consecutive locations for the sorted data tends to be similar. This property lends well to encoding schemes. Thus, even when the error tends to be higher in the non-linear stage, compared with the linear stage, the compression rates are highly similar. For $\tau_\epsilon = 0.1\%$ at each point, the CR lowers to an around 67.6%. This implies that by capturing 99.9% of the original values, the data from the simulation is reduced to less than one-third of its total size.

Figure 4(b) shows the compression ratio (with $\tau_\epsilon = 1\%$) over the entire simulation run using the GTS fusion simulation and Flash astrophysics simulation codes. For GTS Potential data, the compression ratio remains almost the same

across all stages of the simulation. With Flash, after error quantization, most relative errors are 0's. Compressing these values results in negligible storage overhead, and hence CR remains at 80% for the majority of timesteps.

5.3 Effect of Δ–encoding on Index Compression

In this section, we show that compressing along the time dimension further improves ISABELA's overall compression of spatio-temporal scientific datasets by up to 2%—5%. Table 3 show the compression rates achieved for different orders of $\Delta I_{+j}, j = 1, 2, 3$. While increasing W_0 improves spatial compression to a certain extent, it severely diminishes the reduction of the index along the temporal resolution. This is due to the fact that with larger windows and a larger δt between timesteps, the difference in index values lacks the repetitiveness necessary to be compressed well by standard lossless compression libraries.

Table 3. Impact of Δ–encoding on CR for Potential (Density)

W_0	Without Δ–encoding	ΔI_{+1}	ΔI_{+2}	ΔI_{+3}
512	80.08 (80.08)	81.83 (84.14)	81.87 (85.09)	81.68 (85.36)
1024	**81.44 (81.44)**	**83.14 (85.65)**	**83.21 (86.57)**	**82.98 (86.76)**
2,048	81.34 (81.34)	83.03 (85.56)	83.07 (86.44)	82.88 (86.66)
4,096	80.51 (80.51)	82.14 (84.64)	82.21 (85.51)	82.03 (85.76)
8,192	79.32 (79.32)	80.99 (83.38)	81.04 (84.24)	80.83 (84.46)

5.4 Compression Time

The overhead induced on the runtime of the simulation due to *in-situ* data compression is the net sum of the times taken to sort D, build I_π, and perform cubic $B-$spline fitting. However, for a fixed window size W_0, sorting and building the index is computationally less expensive compared to $B-$spline fitting. When executed in serial, ISABELA compresses data at ≈12 MB/s rate, the same as gzip, compression level 6, as shown in Table 2. Within the context of the running simulation, each core is expected to generate around 10 MB of data every 10 seconds that can be reduced to ≈2 MB in 0.83 seconds using ISABELA. Additionaly, to further reduce the impact of *in-situ* compression on the main computation, ISABELA can be executed at the I/O nodes rather than at the compute nodes [1,18].

In the case of compression, parallelization is achieved by compressing each window independently. However, a more fine-grain parallelization can be applied to decompression, as each point in the $B-$spline curve can be reconstructed independently. To evaluate the scalability and parallelization characteristics of ISABELA decompression, we evaluate the time taken for decompression against serial, OpenMP and GPU-based implementations in a single node environment. Figure 5 shows the performance of decompression of all three implementations. The serial implementation is faster when decompressing less than 1,000 points,

but as the number of decompressed points increases, both GPU and OpenMP versions offer the advantage in terms of computational time. This is especially true with a GPU-based implementation, which is better suited for fine-grain parallel computation.

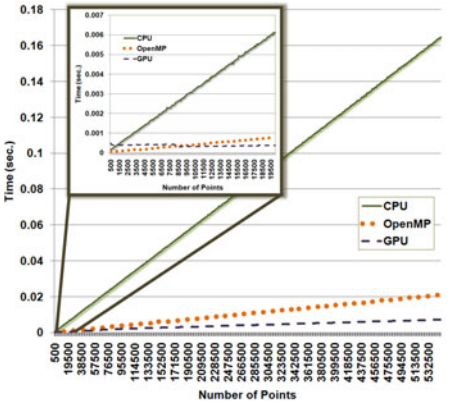

Fig. 5. Computational time for serial, CPU-parallelized, and GPU-parallelized versions of ISABELA's $B-$spline reconstruction part. 8 OpenMP threads were used in this particular plot, corresponding to two quad core Intel Xeon X5355 processors.

5.5 Performance for Fixed Compression

In this section, we evaluate the performance of ISABELA and Wavetlets on 13 public scientific datasets (from numerical simulations, observations, and parallel messages) [3] for the fixed $CR = 81\%$. We compare the averages of ρ_a and $NRMSE_a$ of ISABELA and Wavelets across 400 windows (see Table 4). Out of the 13 datasets, eleven (three) datasets exhibit $\rho_a = 0.98$ with ISABELA (Wavelets). The $NRMSE_a$ values for Wavelets are consistently an order of magnitude higher than for ISABELA. Wavelets outperform ISABELA on `obs_spitzer` consisting of a large number of piecewise linear segments for most of its windows. Cubic $B-$splines do not estimate well when segments are linear.

6 Related Work

Lossy compression methods based spline fitting or Wavelets have been primarily used in the field of visualization, geometric modeling, and signal processing. Very few studies applied such techniques when neither spatial nor temporal correlation of data can be directly exploited. Chou et al. [4] and Lee et al. [9] explored spline fitting for random data to optimize the location of control points to reduce approximation error. In contrast, we aim to transform the data to take advantage of the accuracy and easily-expressible qualities of splines.

Table 4. ISABELA vs. Wavelets for fixed $CR = 81\%$ and $W_0 = 1,024$.

	ρ_a		$NRMSE_a$	
	Wavelets	ISABELA	Wavelets	ISABELA
msg_sppm	0.400 ± 0.287	**0.982 ± 0.017**	0.203 ± 0.142	**0.051 ± 0.015**
msg_bt	0.754 ± 0.371	**0.981 ± 0.054**	0.112 ± 0.151	**0.038 ± 0.024**
msg_lu	0.079 ± 0.187	**0.985 ± 0.031**	0.422 ± 0.103	**0.048 ± 0.015**
msg_sp	0.392 ± 0.440	**0.967 ± 0.051**	0.307 ± 0.243	**0.064 ± 0.033**
msg_sweep3d	0.952 ± 0.070	**0.998 ± 0.006**	0.075 ± 0.036	**0.004 ± 0.003**
num_brain	**0.994 ± 0.008**	0.983 ± 0.028	**0.010 ± 0.011**	0.011 ± 0.005
num_comet	0.988 ± 0.018	**0.994 ± 0.025**	0.020 ± 0.020	**0.010 ± 0.006**
num_control	0.614 ± 0.219	**0.993 ± 0.017**	0.083 ± 0.037	**0.009 ± 0.002**
num_plasma	0.605 ± 0.062	**0.994 ± 0.004**	0.277 ± 0.038	**0.033 ± 0.004**
obs_error	0.278 ± 0.203	**0.994 ± 0.004**	0.303 ± 0.091	**0.024 ± 0.009**
obs_info	0.717 ± 0.136	**0.993 ± 0.006**	0.181 ± 0.078	**0.026 ± 0.016**
obs_spitzer	**0.992 ± 0.001**	0.742 ± 0.004	**0.005 ± 0.000**	0.030 ± 0.000
obs_temp	0.611 ± 0.114	**0.994 ± 0.011**	0.096 ± 0.025	**0.009 ± 0.003**

Lossless compression techniques [3,8,10,13] have been recently applied to floating point data. Unlike most lossless compression algorithms, the techniques presented in [3,10] are specifically designed for fast online compression of data. Lindstrom and Isenberg [10] introduced a method for compressing floating-point values of 2D and 3D grids that functions by predicting each floating point value in the grid and recording the difference between the predictive estimator and the actual data value. They also provide the option of discarding least significant bits of the delta and making the compresion lossy. However, the number of significant precision bits that can be saved is limited to 16, 32, 48, or 64 for double precision data. When applied to a one-dimensional data from GTS simulation, storing only 16 significant bits provided a compression of 82%, which is comparable with ISABELA's, but more than 75% of points had per-point relative error of over 1%. By storing 32 bits, the per-point relative error was found to be within 0.1%, but the compression rate achieved (58.2%) was 13% less than ISABELA's. Morever, like other lossless algorithms, location- specific decoding is not possible.

Most lossy compression techniques either use Wavelets or some form of data quantization to compress the data sets that are fed as input to visualization tools. However, visualization community focuses on providing multi-resolution view-dependent level of detail. The error rate tolerated with lossy compression techniques on data used for visualization tend to be higher when compared to the data used for analysis. Hence, very little work exists that accurately compresses non-image or seemingly random data, even outside the scientific community. In fact, to the best of our knowledge, ISABELA is the first approach to use $B-$spline fitting in the context of reduction for data that is essentially random.

7 Summary

This paper describes ISABELA, an effective *in-situ* method designed to compress spatio-temporal scientific data. ISABELA compresses data over both the spatial and temporal resolutions. For the former, it essentially applies data sorting, as a pre-conditioner, that significantly improves the efficacy of cubic B-spline spatial compression. For the latter, it uses Δ–encoding of the higher-order differences in index values to further reduce index storage requirements. By capturing the relative per point errors and applying error quantization, ISABELA provides over 75% compression on data from GTS, while ensuring 99% accuracy on all values. On 13 other scientific datasets ISABELA provides excellent approximation and reduction, consistently outperforming extensively used Wavelets compression.

Acknowledgements. This work was supported in part by the U.S. Department of Energy, Office of Science (SciDAC SDM Center, DE-AC02-06CH11357, DE-FC02-10ER26002/DE-SC0004935, DE-FOA-0000256, DE-FOA-0000257) and the U.S. National Science Foundation (CCF-1029711 (Expeditions in Computing)). Oak Ridge National Laboratory is managed by UT-Battelle for the LLC U.S. D.O.E. under contract no. DEAC05-00OR22725.

References

1. Abbasi, H., Lofstead, J., Zheng, F., Klasky, S., Schwan, K., Wolf, M.: Extending I/O through high performance data services. In: Cluster Computing, Austin, TX, IEEE International (September 2007)
2. De Boor, C.: A Practical Guide to Splines. Springer, Heidelberg (1978)
3. Burtscher, M., Ratanaworabhan, P.: FPC: A high-speed compressor for double-precision floating-point data (2009), `http://www.csl.cornell.edu/~burtscher/research/FPC/`
4. Chou, J., Piegl, L.: Data reduction using cubic rational B-splines. IEEE Comput. Graph. Appl. 12, 60–68 (1992)
5. Cover, T.M., Thomas, J.: Elements of information theory. Wiley-Interscience, New York (1991)
6. Frazier, M.W.: An introduction to Wavelets through linear algebra, p. 501. Springer, Heidelberg (1999)
7. He, X., Shi, P.: Monotone B-spline smoothing. Journal of the American Statistical Association 93(442), 643–650 (1998)
8. Isenburg, M., Lindstrom, P., Snoeyink, J.: Lossless compression of predicted floating-point geometry. Computer-Aided Design 37(8), 869–877 (2005); CAD 2004 Special Issue: Modelling and Geometry Representations for CAD
9. Lee, S., Wolberg, G., Shin, S.Y.: Scattered data interpolation with multilevel B-splines. IEEE Trans. on Viz. and Comp. Graphics 3(3), 228–244 (1997)
10. Lindstrom, P., Isenburg, M.: Fast and efficient compression of floating-point data. IEEE Trans. on Viz. and Comp. Graphics 12(5), 1245–1250 (2006)
11. Lofstead, J., Zheng, F., Klasky, S., Schwan, K.: Adaptable, metadata rich IO methods for portable high performance IO. In: IPDPS 2009, Rome, Italy (May 2009)

12. Ma, K., Wang, C., Yu, H., Tikhonova, A.: In-situ processing and visualization for ultrascale simulations. Journal of Physics: Conference Series 78(1), 012043 (2007)
13. Ratanaworabhan, P., Ke, J., Burtscher, M.: Fast lossless compression of scientific floating-point data. In: Proc. of the DCC (2006)
14. Sayood, K.: Introduction to data compression. Morgan Kaufmann Publishers Inc., San Francisco (1996)
15. Wang, W.X., al, e.: Gyro-kinetic simulation of global turbulent transport properties in Tokamak experiments. Physics of Plasmas 13(9), 092505 (2006)
16. Welch, T.A.: A technique for high-performance data compression. Computer 17, 8–19 (1984)
17. Wold, S.: Spline functions in data analysis. American Statistical Association and American Society for Quality 16(1), 1–11 (1974)
18. Zheng, F., al, e.: PreDatA–preparatory data analytics on peta-scale machines. In: IPDPS, Atlanta, GA (April 2010)