CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2013; **25**:524–540
Published online 11 July 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2887

SPECIAL ISSUE PAPER

ISABELA for effective *in situ compression of scientific data*

Sriram Lakshminarasimhan [1,2], Neil Shah [1], Stephane Ethier [3], Seung-Hoe Ku [3], C. S. Chang [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*

SUMMARY

Exploding dataset sizes from extreme-scale scientific simulations necessitates efficient data management and reduction schemes to mitigate I/O costs. With the discrepancy between I/O bandwidth and computational power, scientists are forced to capture data infrequently, thereby making data collection an inherently *lossy* process. Although data compression can be an effective solution, the random nature of real-valued scientific datasets renders *lossless* compression routines ineffective. These techniques also impose significant overhead during decompression, making them unsuitable for data analysis and visualization, which 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 pre-conditioner* to seemingly random and noisy data along spatial resolution to achieve an accurate fitting model that guarantees a $\geqslant 0.99$ 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, in terms of data reduction and accuracy.

We extend upon our previous paper by additionally building a communication-free, scalable parallel storage framework on top of ISABELA-compressed data that is ideally suited for extreme-scale analytical processing. The basis for our storage framework is an inherently local decompression method (it need not decode the entire data), which allows for random access decompression and low-overhead task division that can be exploited over heterogeneous architectures. Furthermore, analytical operations such as correlation and query processing run quickly and accurately over data in the compressed space. Copyright © 2012 John Wiley & Sons, Ltd.

Received 22 November 2011; Revised 25 May 2012; Accepted 2 June 2012

KEY WORDS: 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 sth timestep. This process leads to an inherently *lossy* data reduction.

*Correspondence to: Nagiza F. Samatova,, North Carolina State University, Raleigh, NC 27695, USA.
†E-mail: samatova@csc.ncsu.edu

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 [1]. To complement existing approaches, we introduce an effective method for In situ Sort-And-B-spline Error-bounded Lossy Abatement (ISABELA) of scientific data [2]. 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 incompressible [3]. In fact, the majority of the *lossless* compression techniques [4, 5] are not only computationally intensive and therefore hardly suitable for *in situ* processing but also are unable to reduce such data by more than 10% of its original size (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 carried out in curve fitting [6, 7] has 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 in many instances. Hence, this property of monotonic inheritance across temporal resolution offers yet another venue for improvement of the overall data compression ratio.

Although intuitively simple, ISABELA has addressed a number of technical challenges imposed by end-user 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. Because curve fitting accuracy is often data-dependent, ISABELA must be robust in its approximation. Although 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; re-ordering the data requires keeping track of the new position indices to associate the decoded data with its original ordering. Although 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.

In this paper, we extend the work published in an earlier conference [2]. Additional material in this paper discusses parallel compression with ISABELA and the overall data workflow behind the compression engine. A more comprehensive analysis appears here, with the effect of window size on accuracy and benchmarks on datasets from additional petascale simulations. Furthermore, we explore a related research question on the quantitative, visual, and qualitative impact of performing query-driven analytics over ISABELA-compressed data.

2. A MOTIVATING EXAMPLE

Much of the work for *in situ* data reduction in this paper stems from particle simulation codes, specifically Gyrokinetic Tokamak Simulation (GTS) [8] and X-point include Gyrokinetic Code (XGC1) [9] which respectively simulate microturbulence of magnetically confined fusion plasmas of toroidal devices in cores and edges of fusion reactors.

Production runs of these simulation applications typically consume hundreds of thousands of cores on petaflop systems, such as NCCS/ORNL Jaguar [10], utilizing the ADIOS library [11] for performing I/O intensive operations.

Simulation data sets can be broadly divided into: (i) checkpoint data to restart the simulation in case of an execution failure (C&R); (ii) analysis (A) data, such as density and potential fluctuations, for performing various post-processing physics analyses; and (iii) diagnostics data used, for example, for code validation and verification (V&V) (see Table I).

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

DOI: 10.1002/cpe

Table I. Summary of Gyrokinetic Tokamak simulation output data by different categories.

Categoryt	Write frequency	Read access	Size/write	Total size
C&R	Every 1–2 h	Once or never	A few TBs	≈TBs
A	Every $10th$ time step	Many times	A few GBs	≈TBs
V&V	Every $2nd$ time step	A few times	A few MBs	≈GBs

C&R, case of an execution failure; A, analysis; V&V, validation and verification.

main focus of ISABELA. For illustrative purposes, in the rest of the paper, we will use linearized 64-bit double precision floating point arrays. The temporal snapshots consist of series of 172,111 values organized one-dimensionally of both *Potential* and *Density* fluctuations from GTS analysis data, a series of 4096 values organized three-dimensionally from flash astrophysics data, and a series of 124,701 turbulence intensity values organized one-dimensionally from XGC1 analysis data.

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 incompressible [12, 13]). Thus, applying standard *lossless* compression methods do not result in an appreciable data reduction.

Table II illustrates the compression rates achieved and the time required to compress and decode 12,836 KB 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 [14] 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 Equation (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 Equations (2) and (3)

$$\rho(D) = \frac{covariance(D, D')}{std\_dev(D)std\_dev(D')} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\left(d_i - \bar{D}\right)\left(d'_i - \bar{D'}\right)}{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left(d_i - \bar{D}\right)^2}\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left(d'_i - \bar{D'}\right)^2}}, \tag{2}$$

Table II. 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

FPC is a floating-point compressor [5], and LZMA, BZ, ZIP here, represent implementations of Lempel–Ziv–Markov chain, DEFLATE, Burrows-Wheeler algorithms respectively.
The compression method with the highest compression rate is indicated in bold.
*CR achieved by lossy models for 0.99 correlation and 0.01 normalized root mean standard error fixed accuracy.
All runs are performed on an Intel Core 2 Duo 2.2-GHz processor with 4-GB RAM, openSUSE Linux v11.3.

DOI: 10.1002/cpe

$$\text{where } \bar{D} = \frac{\sum_{i=0}^{N-1} d_i}{N}, \text{ and } \bar{D}' = \frac{\sum_{i=0}^{N-1} d_i'}{N}$$

$$NRMSE_M(D) = \frac{RMSE_M(D, D')}{Range(D)} = \frac{\sqrt{\sum_{i=0}^{N-1} (d_i - d_i')^2}}{max(D) - min(D)} \tag{3}$$

Capturing both normalized root mean standard error (NRMSE) and ρ provides not only the extent of error but also measures the degree to which the original and approximated data are linearly related. Thus, achieving $NRMSE_M(D) \sim 0$, $\rho(D) \sim 1$ and $CR \sim 100\%$ would indicate ideal performance.

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, which is, obviously, not a viable option. Applying lossy compression techniques on this data across the *spatial resolution* requires a significant trade-off 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.

4.1. Sorting-based data transformation

Sorting changes the data distribution in the spatial domain from a highly irregular signal (Figure 1(A)) to a smooth and monotonous curve (Figure 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 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 [15] operates on sorted versus unsorted data, respectively.

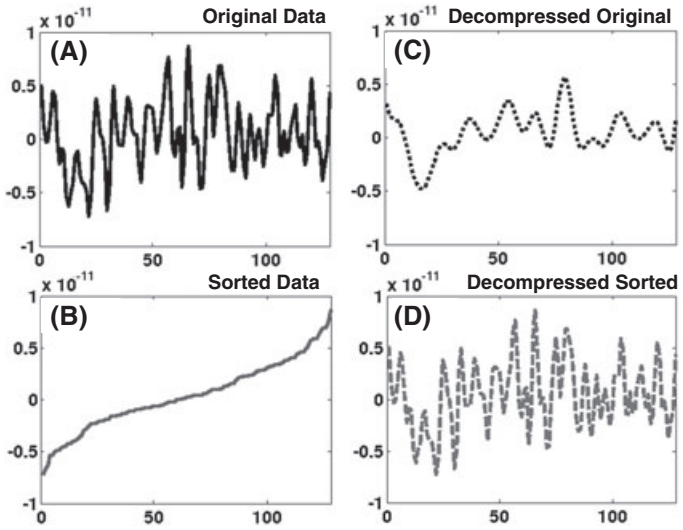

Figure 1. A slice of GTS potential: (A) original; (B) sorted; (C) decoded after B-splines fitting to original; and (D) decoded after B-splines fitting to sorted.

DOI: 10.1002/cpe

4.2. Background: 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 nonlinear 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 spline curve fitting. A *spline curve* is a sequence of curve segments joined together via *knots* that produce a single continuous curve. Each piece of the curve segment is then defined via a lower-order polynomial function. The use of several lower-order polynomial functions to fit smaller regions of the curve tends to perform highly efficiently when compared with using a higher-order polynomial function to fit the entire curve.

A B-spline curve is a sequence of piecewise parametric curves. A *cubic* B-spline is composed of polynomial functions of degree $d = 3$, which has faster interpolation time and produce 'smooth' curves (i.e., second-order differentiable) at the knot locations. Knot locations are points in the parameter space that describe the start and end of a curve segment.

As an example, consider the cubic B-spline curve in Figure 2 with six control points, $P_1, \ldots, P_6$, and 10 knots, $u_1, \ldots, u_{10}$, $u_1 = \cdots = u_4 = 0$ and $u_7 = \ldots = u_{10} = 1.0$. The knot points are the end-points of the piecewise curve segments, S_1, S_2, and S_3, given by $S_1 = \widehat{s_1 s_2}$, $S_2 = \widehat{s_2 s_3}$, and $S_3 = \widehat{s_3 s_4}$, each defined in its parameter space, $u \in [0, 1]$. The control points P define the shape of the curve.

More formally, given a set of m control points $P = \{P_1, P_2, \ldots, P_m\}$ and a knot vector U with its sequence of knots, $u_1 \leqslant u_2 \leqslant \ldots \leqslant u_k$, $k = m + d + 1$, $\forall u_i \in [0, 1]$, the B-spline parametric curve, $S : [u_1, u_k] \to \mathbb{R}^2$, is defined by a linear combination of its basis functions $B_{i,j}(u)$ via Equation (4)

$$S(u) = \sum_{i=1}^{m} B_{i,d}(u) P_i, \text{ where} \tag{4}$$

$$B_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \leqslant u < u_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$B_{i,j}(u) = \frac{u - u_i}{u_{i+j} - u_i} B_{i,j-1}(u) + \frac{u_{i+j+1} - u}{u_{i+j+1} - u_{i+1}} B_{i+1,j-1}(u), \tag{5}$$

where $j = \overline{1, d}$ and $i = \overline{1, m}$, and the basis functions $B_{i,j}(u)$ determine the extent to which the control points P affect the shape of the curve. Thus, 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

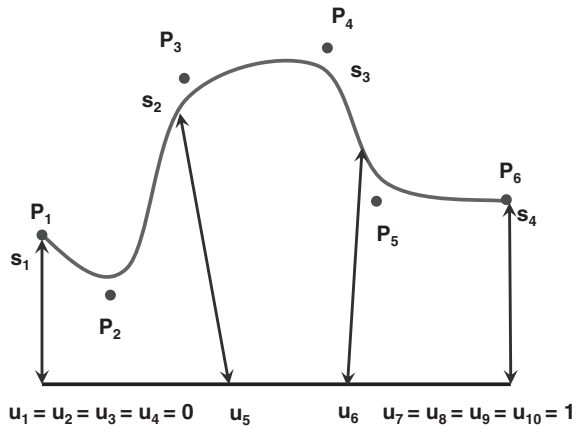

Figure 2. A cubic B-spline fitting with $m = 6$ control points, $k = 10$ knots, and with three piecewise cubic segments.

DOI: 10.1002/cpe

decompressing all the data. Although sorting does re-arrange 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 for 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 Figure 1(A)) requires $C \sim N$ to provide accurate lossy compression and hence, leads to a poor compression rate (see Table II, last column). However, because of the nature of scientific datasets, applying B-splines interpolation after sorting D requires only a few constants, $C = O(1) << N$, in order to provide high decompression accuracy (see Figure 1(D)).

Although significantly reducing the number of B-splines constants C, sorting D will re-order 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 $d_{i_j} \leqslant d_{i_{j+1}}$, $\forall i_j \in I_\pi$. 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. Because each index value i_j requires $log_2 N$ bits, the total storage requirement for I_π is $|I_\pi| = N \times log_2 N$ 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 (rounding up the size of the tail window for the simplicity of analysis), or $D = \bigcup D^k$, $D^i \cap D^j = \emptyset$, $I^k = \{(k-1)W_0, (k-1)W_0 + 1, \ldots, kW_0 - 1\}$, $i, j, k \in \overline{1, N_W}$, $i \neq j$, 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 Equation (6)

$$\begin{aligned} |D_{ISABELA}| &= \sum_{k=1}^{N_W} \left(|D_B^k| + |I_\pi^k| \right), \\ &= N_W \times (C \times 64 + W_0 \times log_2 W_0) \end{aligned} \tag{6}$$

Substituting Equation (6) into Equation (1) and simplifying the resulting equation, we obtain the following compression ratio for ISABELA defined by Equation (7)

$$CR_{\text{ISABELA}}(D) = \left(1 - \frac{log_2(W_0)}{64} - \frac{C}{W_0} \right) \times 100\% \tag{7}$$

From Equation (7), 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. By fixing $W_0 = 1024$, we balance the computational cost of sorting windows and the storage cost for recording the index and the fitting coefficients, which results in an overall compression rate of 81.4% per time step. And empirically, 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, thus balancing for both accuracy and storage.

DOI: 10.1002/cpe

4.4. Error quantization for guaranteed point-by-point accuracy

The aforementioned 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 $D = (d_0, d_1, \ldots, d_{N-1})$ and $D_{ISABELA} = \left(d'_0, d'_1, \ldots, d'_{N-1}\right)$ is defined as $\epsilon_i = \frac{d_i - d'_i}{d_i} \times 100\%$. Although 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 a 32-bit integer 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.

4.4.1. Parallel compression. Fixing the window size and the number of coefficients in a window keeps the compression design embarrassingly parallel, where each window can be compressed independently. However, with the inclusion of error quantization, the size of the encoded errors and hence, the size of the compressed window, no longer remains a constant. To write the compressed data into a contiguous location, each thread must know *a priori* as to where to start writing the B-spline coefficients, index mapping, and the encoded errors. This incurs communication overhead when parallelizing the operation on a per window level. To overcome the delay because of communication, each thread is assigned the task of compressing a fixed number of windows and writing the compressed data including the encoded errors to a local memory space. This data is then written out to disk or passed to I/O libraries by iterating through the local memory of each thread. The number of available threads is usually limited when compared with the number of windows of data; therefore, iterating on a per thread level incurs less overhead than moving the data around for contiguous I/O. In fact, placing compression routines on nodes which produce data has been shown to be effective in reducing the time taken to move the data onto the disk [16]. Given that ISABELA achieves a higher degree of data reduction than the ISOBAR [17] lossless compression technique proposed by Schendel *et al.*, further improvements in writing time can be expected.

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, that is, 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, which is essentially, incompressible. Hence, this index is stored as is. However, 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 Figure 3) because of 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 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.

DOI: 10.1002/cpe

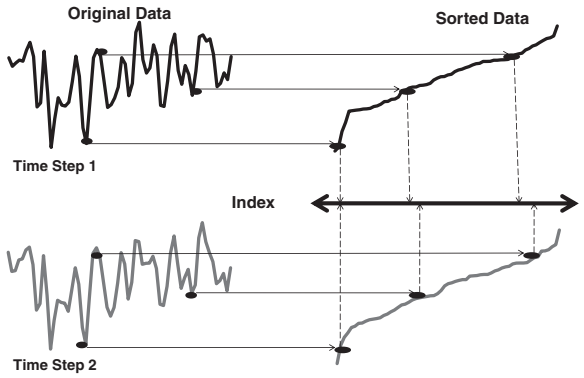

Figure 3. Illustration of Δ encoding of the index across temporal resolution.

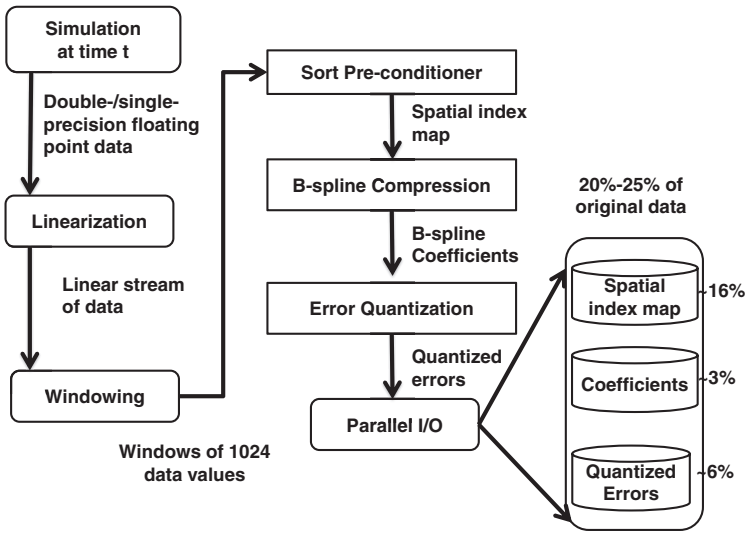

Figure 4. Workflow of ISABELA compression engine from data generation to *in situ* compression to storage.

4.6. ISABELA data workflow

Figure 4 depicts the overall data workflow behind the ISABELA compression engine, starting from data generation to the organization of compressed data in storage. ISABELA compression is characterized by a communication-free model which can be easily parallelized using OpenMP, GPU, and multinode configurations. The parallelism can be extended to the I/O layer as well, thus enabling efficient data analysis. The subsequent section discusses some of the design considerations for each component of the workflow, along with their impacts on storage and accuracy.

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 NRMSE and Pearson correlation (ρ) between the original and decompressed data as accuracy metrics.

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

DOI: 10.1002/cpe

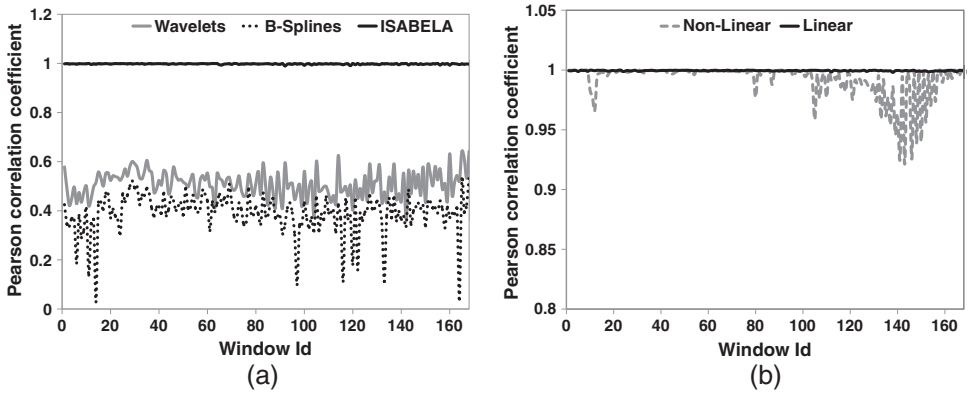

Figure 5. Accuracy (ρ): (a) Per window correlation for wavelets, B-splines, and ISABELA with fixed $CR = 81\%$ for GTS density. (b) Per window correlation for GTS linear and nonlinear stage potential decompressed by ISABELA.

fixed in this experiment: $W_0 = 1024$, $C_{\text{B-spline}} = 150$, and $C_{\text{ISABELA}} = 30$. This fixes $CR = 81\%$, and wavelet coefficients values are thresholded to achieve the same compression rate as well. Figure 5(a) illustrates that ISABELA performs exceptionally well even for much smaller C values because of the monotonic nature of the sorted data. In fact, ρ is > 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.

In-situ Sort-And-B-spline Error-bounded Lossy Abatement performs exceptionally well on data from the linear stages of the GTS simulation (first few thousand timesteps), as shown in Figure 5(b). Yet, the performance for the nonlinear stages (timestep $\approx 10,000$), where the simulation is characterized by a large degree of turbulence is of particular importance to scientists. Figure 5(b), with intentionally magnified correlation values, shows that accuracy for the nonlinear stages across windows drops indeed. Unlike wavelets (Figure 5(a)), however, this correlation does not drop below 0.92.

5.2. *Effect of window size W_0 on accuracy*

In order to keep the compression design embarrassingly parallel, the size of the window W_0 is kept fixed. The choice of window size should not only have a low index storage footprint but also must contain sufficient number of points to approximate the curve accurately. In Figure 6, we calculate

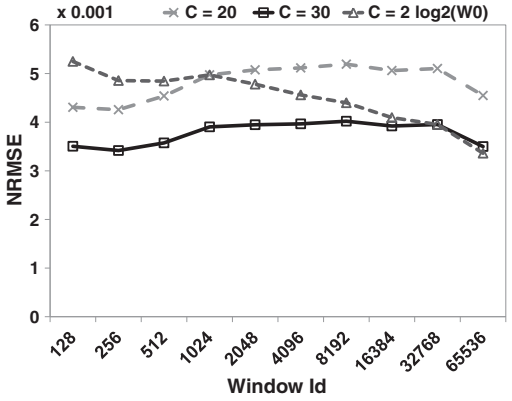

Figure 6. Sensitivity of NRMSE values for ISABELA-compressed GTS potential data across 100 windows over varying window sizes. The compression rate is fixed at $CR = 81.44\%$.

DOI: 10.1002/cpe

the NRMSE sensitivity to different values of constants $C = 20, 30, 2 \times log_2(W_0)$ and to different window sizes W_0 with ISABELA.

One would expect that increasing W_0 without increasing C would increase NRMSE. But this is not the case. As it turns out, the data becomes highly smooth with larger W_0. Hence, NRMSE initially increases, but then it levels off as W_0 keeps growing. When $C = 2 \times log_2(W_0)$, that is, the number of constants is kept constant proportional to the window size, NRMSE decreases as W grows. The choice of $W = 1024$ and $C = 30$ balances the trade-off between accuracy and compression rate, providing a fixed 81% reduction in the size of the data.

5.3. Trade-off between compression and per point accuracy

To alleviate the aforementioned problem, we apply error quantization, as described in Section 4.4. In both linear and nonlinear stages of GTS simulation, the compression ratios are similar when the per point relative error (τ_ϵ) is fixed (see Figure 7(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 nonlinear 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 7(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.

5.4. 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%. For example, on density with $W_0 = 1024$, the index size reduces from 15.63% to 10.63%–13.63% with index compression. Table III shows the compression rates achieved for different orders of $\Delta I_{+j}, j = 1, 2, 3$. Although increasing W_0 improves spatial compression to a certain extent, it severely diminishes the reduction of the index along the temporal resolution. This is because of 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.

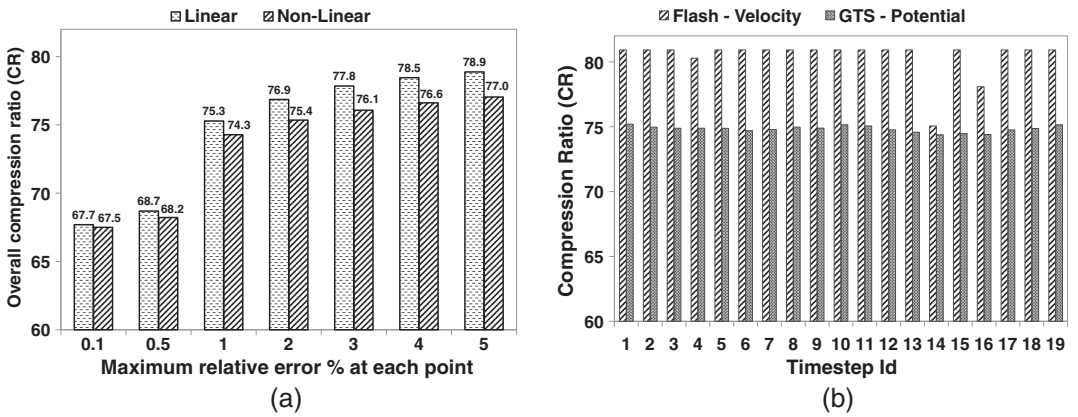

Figure 7. Compression ratio (CR) performance: (a) For various per point relative error thresholds (τ_ϵ) in GTS potential during linear and nonlinear stages of the simulation. (b) For various timesteps with $\tau_\epsilon = 1\%$ at each point (for GTS potential: $t_1 = 1000$, $\Delta t = 1500$; for velocity in flash: $t_1 = 3000$, $\Delta t = 3500$.

DOI: 10.1002/cpe

Table III. 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)**
2048	81.34 (81.34)	83.03 (85.56)	83.07 (86.44)	82.88 (86.66)
4096	80.51 (80.51)	82.14 (84.64)	82.21 (85.51)	82.03 (85.76)
8192	79.32 (79.32)	80.99 (83.38)	81.04 (84.24)	80.83 (84.46)

The window size that results in the best compression rate is indicated in bold.

5.5. Compression time

The overhead induced on the runtime of the simulation because of *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 with B-spline fitting. When executed in serial, ISABELA compresses data at average of around 12 MB/s rate, the same as gzip compression level 6, as shown in Table II. Within the context of the running simulation, if each core generates around 10 MB of data every 10 s, it can be reduced to ≈2 MB in 0.83 s using ISABELA. Additionally, 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 [18, 19].

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 8 shows the performance of decompression of all three implementations. The serial implementation is faster when decompressing less than 1000 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.

5.6. Performance for fixed compression

In this section, we evaluate the performance of ISABELA and wavelets on 13 public scientific datasets (from numerical simulations, observations, and parallel messages) [4] and seven datasets from petascale simulation applications for the fixed $CR = 81\%$. For the wavelet transform, we

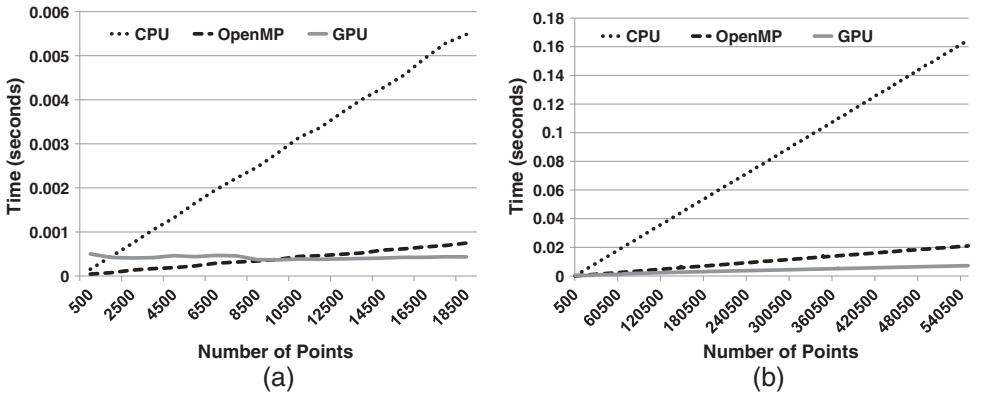

Figure 8. Computational time for serial, CPU-parallelized, and GPU-parallelized versions of ISABELA's B-spline reconstruction part. Plots (a) and (b) compare performance when a small and large number of points are reconstructed, respectively. Two Intel Xeon X5355 quad-core 2.66 Ghz processors equipped with 16 GB of RAM were used for serial execution and additionally, with 16 threads for OpenMP execution. GPU execution utilized the NVIDIA C2050 card with 3 GB memory.

DOI: 10.1002/cpe

Table IV. ISABELA versus wavelets for fixed $CR = 81\%$ and $W_0 = 1024$.

	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**
gts_phi	0.886 ± 0.030	**0.998 ± 0.003**	0.075 ± 0.051	**0.004 ± 0.001**
gts_zion	0.246 ± 0.024	**0.996 ± 0.003**	0.146 ± 0.143	**0.021 ± 0.009**
gts_zeon	0.232 ± 0.039	**0.995 ± 0.003**	0.242 ± 0.063	**0.018 ± 0.005**
xgc_iphase	0.235 ± 0.027	**0.992 ± 0.013**	0.291 ± 0.012	**0.022 ± 0.022**
flash_gamc	0.918 ± 0.063	**0.989 ± 0.010**	0.087 ± 0.057	**0.008 ± 0.007**
flash_velx	0.893 ± 0.055	**0.999 ± 0.005**	0.129 ± 0.059	**0.003 ± 0.002**
flash_vely	0.937 ± 0.054	**0.993 ± 0.008**	0.079 ± 0.047	**0.003 ± 0.003**

Bold values identify the compression method that provides higher correlation, lower NRMSE on each of the datasets.

use the 'fields' library package in R. To fix the compression ratio, the wavelet coefficients with the lowest absolute values are reduced to 0. We then compare the averages of ρ_a and $NRMSE_a$ of ISABELA and wavelets across 400 windows (see Table IV). Out of the 20 datasets, 18 (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`, which consists of a large number of piecewise linear segments for most of its windows. Cubic B-splines do not estimate well when data segments are linear.

5.7. *Performance for fixed accuracy*

From the end-user perspective, the input arguments are defined by accuracy levels. In this section, we evaluate the storage footprint of ISABELA under strict accuracy constraints of $\rho > 0.99$, and

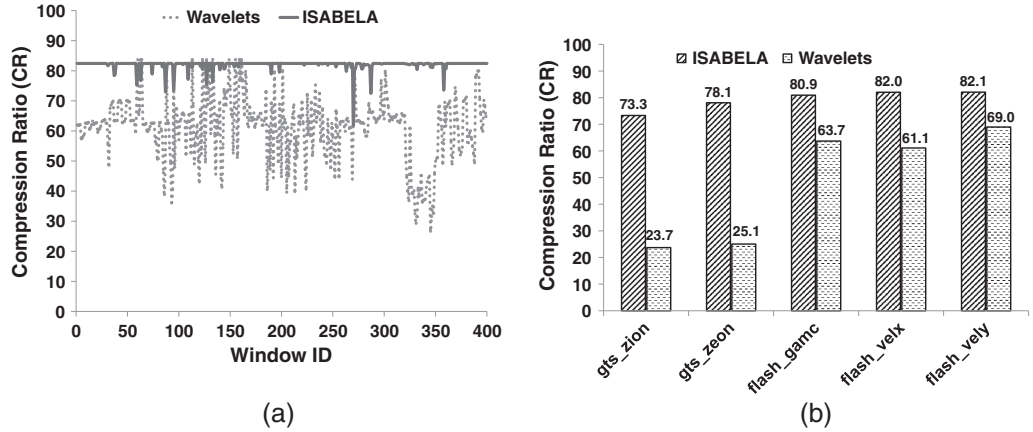

Figure 9. Compression ratio (CR) performance with per window constraint $\rho > 0.99$ and NRMSE< 0.01: (a) On each window, with varying number of coefficients, in flash velx. (b) Overall storage cost over 400 windows across various petascale simulation datasets.

DOI: 10.1002/cpe

NRMSE < 0.01. Figure 9(a) contrasts the storage consumed by wavelets against ISABELA, when the number of coefficients saved per window with both methods are made flexible. In the case of ISABELA, the compression ratio in each window remains close to the fixed values ($C = 30$). Even on hard-to-compress datasets like gts_zion, as seen in Figure 9(b), ISABELA offers 3× reduction compared with wavelets at the same accuracy levels.

5.8. Scientific data analysis on ISABELA-compressed data

Scientific data is qualitatively analyzed using visualization routines and statistically using scripting languages. More often than not, scientists need to explore the data in order to formulate their hypotheses, which are subsequently analyzed using the above routines. To perform exploratory data analysis, large volumes of scientific data need to be queried efficiently and repeatedly. In this section we take a detailed look at how compressing data with ISABELA affects the analysis of scientific data from simulations.

5.8.1. Quantitative analysis. Consider the case of XGC [9], a particle simulation code for analyzing the turbulence in Tokamak edge plasmas in nuclear reactors. One output produced by the simulation is the turbulence of particles across grids that are divided into radial zones. To analyze this data, scientists often look at the time-based correlation of normalized turbulence intensity values across surfaces spanning different radial zones. Table V shows the correlation matrix on average turbulence intensity values between five radial zones. Table VI holds the same correlation matrix but on ISABELA-compressed data. The absolute difference of the derived analysis between ISABELA-compressed and original data is minimal.

5.8.2. Visual analysis. Visualization tools are one of the primary aides of scientific data analysis. As these tools are used for interactive and real-time analysis, they benefit from large-scale data reduction that alleviates the bottleneck because of slow I/O accesses. In this section, we evaluate the accuracy of ISABELA-compressed data on visual analysis. We use the example of the XGC data analysis routine, similar to the scenario mentioned earlier. Apart from calculating the correlation, the normalized turbulence intensity values across different regions of the grid are visualized as time series. Figure 10 compares the plot of turbulence intensity values over 620 timesteps for original and ISABELA-compressed data with $\epsilon = 0.1\%$ error quantization. ISABELA accurately captures the physical phenomena as seen with the original data.

Table V. XGC data analysis showing the correlation between radial zones on the original data.

Zone	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5
1	1	0.64	0.35	0.60	0.82
2	0.64	1	−0.05	0.38	0.86
3	0.35	−0.05	1	0.62	0.05
4	0.60	0.38	0.62	1	0.54
5	0.82	0.86	0.05	0.54	1

Table VI. Impact of error quantization on correlation between radial zones on ISABELA-compressed data, and the difference with the correlation over original data.

Zone	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5
1	1 (0.000)	0.64 (0.053)	0.35 (0.005)	0.60 (0.015)	0.82 (0.074)
2	0.64 (0.053)	1 (0.000)	−0.05 (0.098)	0.38 (0.032)	0.86 (0.071)
3	0.35 (0.005)	−0.05 (0.098)	1 (0.000)	0.62 (0.022)	0.05 (0.005)
4	0.60 (0.015)	0.38 (0.032)	0.62 (0.022)	1 (0.000)	0.54 (0.079)
5	0.82 (0.074)	0.86 (0.071)	0.05 (0.005)	0.54 (0.079)	1 (0.000)

DOI: 10.1002/cpe

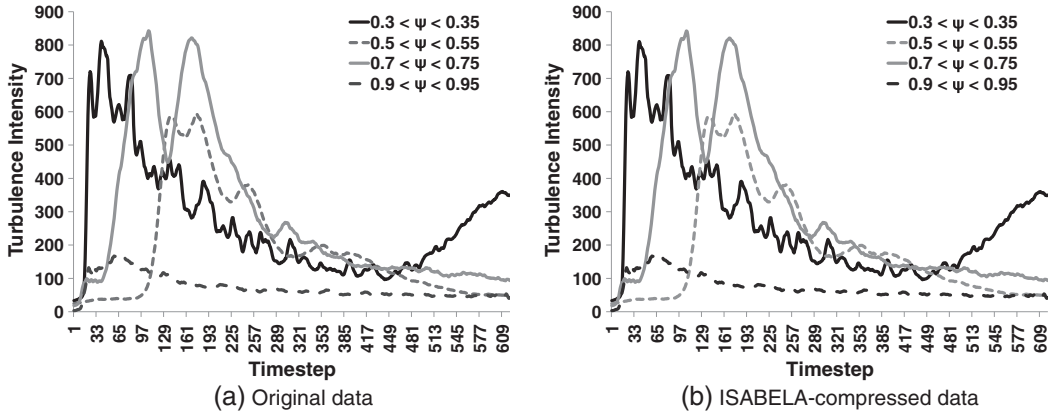

Figure 10. High fidelity visual analysis with ISABELA-compressed data.

5.8.3. Quality of query-driven analysis. Scientific discovery in general is supported by the need to repeatedly query a large amount of data for hypothesis testing or to perform exploratory analytics. An example of a query-driven hypothesis is 'which locations in the grid have temperature $< 25°$ Celsius.' This problem could be solved using database indexing techniques for query processing. However, at the current scale and rate of data production, indices used to accelerate query processing are more voluminous than the actual data. The strain on storage becomes unacceptable and the processing again becomes inefficient because of heavy I/O.

In ISABELA-QA [20], we presented a scalable query processing technique on ISABELA-compressed data that is lightweight in terms of storage and memory footprint, while being efficient for large-scale data analysis. ISABELA-QA delivers an order of magnitude faster response times over state-of-the-art indexing techniques.

In this section, we analyze the quality of data obtained from ISABELA-compressed data. Figure 11 shows the variation of precision in the retrieved result over varying query selectivity sizes. Query selectivity here refers to the ratio of the amount of data returned by the query to the actual data, and precision refers to the number of 'relevant' data points retrieved divided by the total number retrieved. Naturally, when the user-defined relative error parameter (ϵ) is decreased when storing the data, the quality of query results on ISABELA-compressed data becomes more accurate. On a query of the form *variable* $<$ *value*, increasing the selectivity results in precision value ≈ 1 (Figure 11). This is because when the data follows a normal distribution, the absolute value of errors includes a larger number of points at the tails, which results in lower precision or larger false positive rates. To ensure false negatives do not exist, user-defined queries are rewritten on ISABELA-compressed data to include the range $\pm\epsilon$.

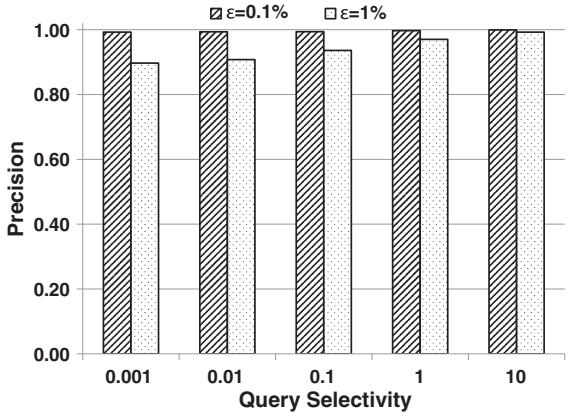

Figure 11. Precision of query results obtained on ISABELA-compressed 3.2-GB GTS potential data.

DOI: 10.1002/cpe

6. RELATED WORK

Lossy compression methods based on 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.* [21] and Lee *et al.* [22] 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.

Extensively used lossy compression techniques like discrete cosine transform (DCT) and discrete wavelet transform (DWT) utilize spatial properties inherent in scientific data to achieve compression. DCT and DWT transform the data into frequency and spatio-frequency domains, respectively, reducing the number of coefficients required to capture the data in the transformed space. Compression is achieved by thresholding away coefficients with small values.

The *robustness* of the sorting pre-conditioner is evident from the aforementioned results obtained using ISABELA, which outperforms transform-based wavelet reduction. As an alternate to B-spline-based reduction after sorting, the use of transform-based techniques to compress monotonic data can be effective as well. In a monotonic set of values, the spatial correlation of data is maximized, and both DCT and DWT can be expected to provide comparable levels of accuracy.

However, even as accuracy levels remain comparable with ISABELA, point-wise decompression will no longer be possible. Query-driven analytics require local decompression in order to be efficient, which is lost when performing multilevel transformations on the data.

Lossless compression techniques [4, 5, 23, 24] have been recently applied to floating point data. Unlike most lossless compression algorithms, the techniques presented in [4, 24] are specifically designed for fast online compression of data. Lindstrom and Isenberg [24] 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 compression 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 the total 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. Moreover, like other lossless algorithms, location specific decoding is not possible.

Other lossy compression techniques use variants of transform-based reduction, followed by some form of data quantization to compress the data sets that are fed as input to visualization tools. However, visualization community focuses on providing multiresolution view-dependent level of detail. The error rate tolerated with lossy compression techniques on data used for visualization tend to be higher when compared with 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. CONCLUSION

This paper describes ISABELA, an effective *in situ* method designed to compress spatio-temporal scientific data and perform analytical operations over the compressed data. The ISABELA compression starts by partitioning the data into small windows and applying a sorting pre-conditioner, which significantly improves the efficacy of cubic B-spline spatial compression. It also exploits temporal patterns prevalent in scientific datasets, applying a Δ encoding of the higher-order differences in index values to further reduce index storage requirements. The indexing scheme built on top of the ISABELA data presents a storage-efficient format that accelerates query processing on heterogeneous architectures.

DOI: 10.1002/cpe

On 20 scientific datasets, ISABELA provides excellent approximation and reduction. Not only does it consistently outperforms wavelet transform technique, but also delivers better performance, in terms of both compression ratio and accuracy. By capturing the relative per point errors and applying error quantization, ISABELA provides over 75% compression on data from XGC, GTS and Flash simulation applications, while ensuring 99% accuracy on almost all values. Furthermore, several analytical operations, such as correlation and query-driven processing benefit from quick approximate solutions that can be obtained by operating over ISABELA-compressed data. The storage-efficient solution over error-bounded compressed data, leads to accurate results on analytical operations over XGC and GTS simulation data sets ($> 99\%$ at $\epsilon = 0.1\%$) when compared with the original data. The ISABELA-compressed data and its parallel storage framework are thus ideally suited for scientific data analytics and visualization routines.

ACKNOWLEDGEMENTS

We would like to thank ORNLs and ANLs leadership class computing facilities, OLCF and ALCF respectively, for the use of their resources. We would also like to acknowledge the use of those scientific datasets at Flash Center for Computational Science. Additionally, we would like to thank the development teams of the XGC1 and GTS simulations for the data used in this paper. This work was supported in part by the US Department of Energy, Office of Science and the US National Science Foundation (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. Ma K, Wang C, Yu H, Tikhonova A. In-situ processing and visualization for ultrascale simulations. *Journal of Physics: Conference Series* 2007; **78**(1):012043. DOI: 10.1088/1742-6596/78/1/012043.
2. Lakshminarasimhan S, Shah N, Ethier S, Klasky S, Latham R, Ross R, Samatova N. Compressing the incompressible with isabela: in-situ reduction of spatio-temporal data. In *Euro-Par 2011 Parallel Processing*, Vol. 6852, Jeannot E, Namyst R, Roman J (eds), Lecture Notes in Computer Science. Springer: Berlin / Heidelberg, 2011; 366–379.
3. Welch T. A technique for high-performance data compression. *Computer* 1984; **17**:8–19. DOI: 10.1109/MC.1984.1659158.
4. Burtscher M, Ratanaworabhan P. FPC: a high-speed compressor for double-precision floating-point data. *IEEE Transactions on Computers* 2009; **58**(1):18–31. DOI: 10.1109/TC.2008.131.
5. Ratanaworabhan P, Ke J, Burtscher M. Fast lossless compression of scientific floating-point data. *Proceedings of the Data Compression Conference (DCC)*, Utah, 2006; 133–142, DOI: 10.1109/DCC.2006.35.
6. He X, Shi P. Monotone B-spline smoothing. *Journal of the American Statistical Association* 1998; **93**(442):643–650. DOI: 10.2307/2670115.
7. Wold S. Spline functions in data analysis. *American Statistical Association and American Society for Quality* 1974; **16**(1):1–11. DOI: 10.2307/1267485.
8. Wang WX, Lin Z, Tang WM, Lee WW, Ethier S, Lewandowski JLV, Rewoldt G, Hahm TS, Manickam J. Gyro-kinetic simulation of global turbulent transport properties in Tokamak experiments. *Physics of Plasmas* 2006; **13**(9):092505. Available from: http://link.aip.org/link/?PHP/13/092505/1.
9. Ku S, Chang CS, Diamond PH. Full-f gyrokinetic particle simulation of centrally heated global ITG turbulence from magnetic axis to edge pedestal top in a realistic Tokamak geometry. *Nuclear Fusion* 2009; **49**(11):115021.
10. Bland A, Kendall R, Kothe D, Rogers J, Shipman G. Jaguar: the world's most powerful computer. *Proceedings of the Cray User Group Conference*, Atlanta, 2009; 11–17.
11. Lofstead J, Zheng F, Klasky S, Schwan K. Adaptable, metadata rich IO methods for portable high performance IO. *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, Rome, Italy, 2009; 1–10, DOI: 10.1109/IPDPS.2009.5161052.
12. Cover TM, Thomas J. *Elements of Information Theory*. Wiley-Interscience: New York, NY, USA, 1991.
13. Sayood K. *Introduction to Data Compression*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1996.
14. Frazier M. *An Introduction to Wavelets Through Linear Algebra*. Springer-Verlag: New York, 1999.
15. De Boor C. *A Practical Guide to Splines*. Springer-Verlag: New York, 1978.
16. Schendel E, Pendse S, Jenkins J, Boyuka D, Gong Z, Lakshminarasimhan S, Liu Q, Kolla H, Chen J, Klasky S, Ross R, Samatova NF. ISOBAR hybrid compression-I/O interleaving for large-scale parallel I/O optimization. *Proceedings of the 21st International Symposium on High Performance Distributed Computing (HPDC)*, Delft, The Netherlands, 2012; 61–72.
17. Schendel E, Jin Y, Shah N, Chen J, Chang CS, Ku S-H, Ethier S, Klasky S, Latham R, Ross R, Samatova NF. ISOBAR pre-conditioner for effective and high-throughput lossless data compression. *IEEE 28th International Conference on Data Engineering (ICDE)*, Washington D.C., 2012; 321–332.

18. Abbasi H, Lofstead J, Zheng F, Schwan K, Wolf M, Klasky S. Extending i/o through high performance data services. *IEEE International Conference on Cluster Computing and Workshops (CLUSTER)*, New Orleans, LA, USA, 2009; 1–10.
19. Zheng F, Abbasi H, Docan C, Lofstead J, Liu Q, Klasky S, Parashar M, Podhorszki N, Schwan K, Wolf M. Predata: preparatory data analytics on peta-scale machines. *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, Atlanta, GA, USA, 2010; 1–12.
20. Lakshminarasimhan S, Jenkins J, Arkatkar I, Gong Z, Kolla H, Ku S-H, Ethier S, Chen J, Chang CS, Klasky S, Latham R, Ross R, Samatova NF. ISABELA-QA: query-driven analytics with ISABELA-compressed extreme-scale scientific data. *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis,* SC, Seattle, WA, USA, 2011; 31:1–31:11, DOI: 10.1145/2063384.2063425.
21. Chou J, Piegl L. Data reduction using cubic rational B-splines. *IEEE Computer Graphics and Applications* 1992; **12**(3):60–68. DOI: 10.1109/38.135914.
22. Lee S, Wolberg G, Shin SY. Scattered data interpolation with multilevel B-splines. *IEEE Transactions on Visualization and Computer Graphics* 1997; **3**(3):228–244.
23. Isenburg M, Lindstrom P, Snoeyink J. Lossless compression of predicted floating-point geometry. *Computer-Aided Design* 2005; **37**(8):869–877. CAD '04 Special Issue: Modelling and Geometry Representations for CAD.
24. Lindstrom P, Isenburg M. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics* 2006; **12**(5):1245–1250. DOI: 10.1109/TVCG.2006.143.