Computers & Geosciences 93 (2016) 45–54

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

Reducing disk storage of full-3D seismic waveform tomography (F3DT) through lossy online compression

Peter Lindstrom [a], Po Chen [b,*], En-Jui Lee [c]

[a] Lawrence Livermore National Laboratory, USA
[b] Department of Geology and Geophysics, University of Wyoming, USA
[c] Department of Earth Sciences, National Cheng Kung University, Taiwan

ARTICLE INFO

Article history:
Received 30 October 2015
Received in revised form
16 April 2016
Accepted 18 April 2016
Available online 5 May 2016

Keywords:
Seismic tomography
Full-3D tomography
Scattering-integral method
Waveform tomography
Full-waveform
Full-wave
Compression
Lossy compression
Online compression

ABSTRACT

Full-3D seismic waveform tomography (F3DT) is the latest seismic tomography technique that can assimilate broadband, multi-component seismic waveform observations into high-resolution 3D subsurface seismic structure models. The main drawback in the current F3DT implementation, in particular the scattering-integral implementation (F3DT-SI), is the high disk storage cost and the associated I/O overhead of archiving the 4D space-time wavefields of the receiver- or source-side strain tensors. The strain tensor fields are needed for computing the data sensitivity kernels, which are used for constructing the Jacobian matrix in the Gauss–Newton optimization algorithm. In this study, we have successfully integrated a lossy compression algorithm into our F3DT-SI workflow to significantly reduce the disk space for storing the strain tensor fields. The compressor supports a user-specified tolerance for bounding the error, and can be integrated into our finite-difference wave-propagation simulation code used for computing the strain fields. The decompressor can be integrated into the kernel calculation code that reads the strain fields from the disk and compute the data sensitivity kernels. During the wave-propagation simulations, we compress the strain fields before writing them to the disk. To compute the data sensitivity kernels, we read the compressed strain fields from the disk and decompress them before using them in kernel calculations. Experiments using a realistic dataset in our California statewide F3DT project have shown that we can reduce the strain-field disk storage by at least an order of magnitude with acceptable loss, and also improve the overall I/O performance of the entire F3DT-SI workflow significantly. The integration of the lossy online compressor may potentially open up the possibilities of the wide adoption of F3DT-SI in routine seismic tomography practices in the near future.

1. Introduction

Seismic tomography has been one of the most effective means for imaging the internal structure of the Earth in the past few decades (e.g., Anderson and Dziewonski, 1984; Nolet, 1987a, 1987b; Iyer, 1993; Nolet, 2008, 2012). The techniques used in seismic tomography have been constantly improving. Recent advances in computing technology has drastically reduced the computational cost for solving the 3D (visco)elastic seismic wave equation, which has enabled full-3D tomography (F3DT) (e.g., Chen et al., 2007a, 2007b; Fichtner et al., 2009; Tape et al., 2010; Lee et al., 2014a, 2014b; Chen and Lee, 2015). In F3DT, the starting seismic structural model can be fully three-dimensional and the Fréchet (sensitivity) kernels are computed by numerically solving the inhomogeneous equations of motion for a heterogeneous, (an) elastic solid. It accounts for the nonlinearity of waveform inversions through iterated cycles of wave-propagation simulations, misfit measurements, sensitivity kernel calculations and inversions.

There are two complementary F3DT implementations: the adjoint-wave-field method (F3DT-AW), which constructs the gradient of the objective function using the adjoint method and solves the optimization problem using gradient-based algorithms (e.g., Fichtner et al., 2009; Tape et al., 2010), and the scattering-integral method (F3DT-SI), which sets up the Jacobian of the objective function by calculating and storing the data sensitivity (Fréchet) kernel for each misfit measurement and solves the optimization problem using the Gauss–Newton algorithm (e.g., Chen et al., 2007a; Lee et al., 2014a). These two types of implementations are based on the same physics, but their computational requirements can be highly different (Chen et al., 2007b). For tomography problems involving a large number of seismic sources F3DT-SI may significantly reduce the total amount of computing time at the expense of substantially higher disk storage cost. For

* Corresponding author.
E-mail address: pchen@uwyo.edu (P. Chen).

http://dx.doi.org/10.1016/j.cageo.2016.04.009
0098-3004/

the F3DT inversion in Southern California (Lee et al., 2014a), the peak disk storage for F3DT-SI was about 39 TB, while the peak disk storage for F3DT-AW was only about 200 GB, a nearly 200 times difference. The high disk storage cost of F3DT-SI is becoming the main obstacle to the wide adoption of F3DT-SI in routine seismic tomography, especially on small to medium-sized shared computer clusters without large amounts of high-speed disk storage.

In this paper, we describe a potential solution for significantly reducing the disk storage of F3DT-SI through lossy but error-bounded online compression. Our compression algorithm, named *zfp*, provides high compression ratios with minimal CPU overhead and can work inside the wave-propagation simulation code and the sensitivity kernel calculation code in a streaming setting during the I/O stage (Lindstrom, 2014). Whereas *zfp* was originally designed for fixed-rate compression in order to support random access, we use its fixed-accuracy (variable-rate) mode in order to limit compression-induced errors. Although this mode sacrifices the ability to perform constant-time random access, we require only sequential reads and writes of entire strain fields. Moreover, relaxing the fixed-rate constraint can significantly improve the quality per bit of compressed storage. Preliminary experiments using realistic simulations in our California statewide F3DT-SI inversion show highly promising results. On average, the disk space for storing the 4D strain tensor fields can be reduced by at least an order of magnitude with much improved I/O performance. The kernels computed from the compressed strain fields have negligible differences from those computed using the raw strain fields. By integrating *zfp*, we expect to make F3DT-SI much more affordable on small clusters.

2. Disk storage cost of F3DT-SI

F3DT is often implemented using gradient- or Hessian-based iterative optimization algorithms. The discretized earth structural model $\mathbf{m}$ is iteratively updated through a finite series of perturbations,

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \Delta\mathbf{m}_k, \quad k = 0, 1, 2, \ldots, K \tag{1}$$

where k is the iteration index. The perturbation for the kth iteration, $\Delta\mathbf{m}_k$, can be obtained by minimizing an objective function,

$$\chi^2(\mathbf{m}, \mathbf{m}_k) = \mathbf{d}^{\mathrm{T}}(\mathbf{m}, \mathbf{m}_k)\mathbf{C}_d^{-1}\mathbf{d}(\mathbf{m}, \mathbf{m}_k) + (\mathbf{m} - \mathbf{m}_k)^{\mathrm{T}}\mathbf{C}_m^{-1}(\mathbf{m} - \mathbf{m}_k) \tag{2}$$

where $\mathbf{m}$ is the "target" structural model, $\mathbf{d}(\mathbf{m}, \mathbf{m}_k)$ is a column-vector composed of misfit measurements that quantify the discrepancies between the ith-component observed seismogram generated by the sth seismic source and recorded at the rth receiver, $\bar{u}_i^s(\mathbf{x}_r, t)$, and the corresponding synthetic seismogram $u_i^s(\mathbf{x}_r, t)$ computed using the latest structural model $\mathbf{m}_k$, $\mathbf{C}_d$ and $\mathbf{C}_m$ are respectively the data and model covariance matrices.

In F3DT-SI, the objective function in Eq. (2) is minimized using the Gauss–Newton algorithm, which requires the solution of the Gauss–Newton normal equation

$$\begin{bmatrix} \mathbf{C}_d^{-1/2}\mathbf{A}_k \\ \mathbf{C}_m^{-1/2} \end{bmatrix} \Delta\mathbf{m}_k = \begin{bmatrix} \mathbf{C}_d^{-1/2}\mathbf{d}_k \\ \mathbf{0} \end{bmatrix} \tag{3}$$

where $\mathbf{A}_k = \partial\mathbf{d}_k/\partial\mathbf{m}_k$ is the Jacobian matrix for the kth iteration. In F3DT-SI, the Jacobian matrix is explicitly constructed and Eq. (3) is solved using the scalable parallel LSQR algorithm (Lee et al., 2013).

Each row of the Jacobian is a discretized data sensitivity kernel, which can be computed using the 4D strain fields from the source and those from the receiver. Equations for constructing the data sensitivity kernels using the receiver-side strain Green's tensors (RSGTs) have been given in (e.g., Zhao et al., 2005', 2006; Chen et al., 2007a, 2007b; Chen and Lee, 2015). The calculation involves temporal convolution between the strain field from the source and the RSGT for the corresponding receiver, which can be computed by placing a point impulsive source at the receiver location (Zhao et al., 2006). To construct the kernels for all misfit measurements, we need to store either the RSGTs or the source-side strain fields. When seismic sources outnumber receivers, it is more economical to store the RSGTs.

In practice, the kernels are usually smoother than the strain fields and we often regularize the inverted model perturbation $\Delta\mathbf{m}_k$ through smoothness damping. Therefore we can sample the kernels on a mesh that is sparser than the mesh used for the wave-propagation simulations. The accuracy of the temporal convolution is usually sufficient if we have 10 time samples per dominant period. Because of these considerations, the disk space for storing strain fields can be reduced significantly through decimation in space and time. But even after decimation, the disk storage for all strain fields used in a realistic inversion can still be significant.

3. Lossy online compression of strain fields

Previous studies on seismic data compression mainly focused on the compression of observed active-source seismic data in the space-time domain in an off-line setting (e.g., Wood, 1974; Jonsson and Spanias, 1990; Mandyam et al., 1996; Villasenor et al., 1996; Wang and Wu, 2000; Averbuch et al., 2001). Blind application of traditional lossless compression algorithms on observed active-source seismic wavefields can only provide low compression ratios of around 2 (e.g., Villasenor et al., 1996), while applications of lossy compression algorithms were able to achieve compression ratios ranging from ~20 to over 100 with acceptable losses of useful seismic information (e.g., Villasenor et al., 1996; Wang and Wu, 2000; Averbuch et al., 2001). For the 4D synthetic RSGTs from the California statewide inversion considered in this study, perfectly lossless compression using a state-of-the-art floating-point lossless compressor *FPZIP* (Lindstrom and Isenburg, 2006) provided a compression ratio of merely 1.55. For F3DT purposes, a lossless compression of synthetic strain fields is both unnecessary and inefficient. A more desirable compression scheme is a lossy algorithm that can work in an online I/O setting from within the wave-propagation simulation code with minimal CPU overhead and can achieve significant compression ratios without introducing significant artifacts into the data sensitivity kernels.

Lossy compression of observed active-source seismic data has been extensively studied in the past two decades (e.g., Lervik et al., 1996; Villasenor et al., 1996; Vassiliou and Wickerhouser, 1997; Wang and Wu, 2000; Averbuch et al., 2001; Al-Moohimeed, 2004; Wang et al., 2004; Aparna and David, 2006; Wu et al., 2006; Xie and Qin, 2009; Aqrawi and Elster, 2011; Zheng and Liu, 2012; Fajardo et al., 2015). The majority of the compression algorithms usually follow a 3-stage process: de-correlating transformation, quantization and coding. In the transformation stage, suitable basis functions can lead to a much sparser representation of the original data in the transformed domain. Wavelets, wavelet packets, (adaptive) local trigonometric functions and various combinations of the above have been widely used in previous studies (e.g., Villasenor et al., 1996; Al-Moohimeed, 2004; Wang and Wu, 2000; Wu et al., 2006; Zheng and Liu, 2012). The floating-point transform coefficients are then mapped to a set of integers in the quantization stage. The majority of previous studies adopted uniform quantization schemes (e.g., Lervik et al., 1996; Al-Moohimeed, 2004; Aparna and David, 2006; Wu et al., 2006; Fajardo et al., 2015). In general, as the number of quantization bits decreases, the compression ratio, as well as the loss of useful

information, increases. In the final stage, the quantized transform coefficients are encoded using a lossless coding scheme, usually an entropy encoding technique such as Huffman coding (e.g., Villasenor et al., 1996; Vassiliou and Wickerhouser, 1997; Wang et al., 2004; Aqrawi and Elster, 2011; Fajardo et al., 2015) or arithmetic coding (e.g., Lervik et al., 1996; Averbuch et al., 2001; Wu et al., 2006; Xie and Qin, 2009; Fajardo et al., 2015). The primary goal in previous studies was to increase the compression ratio while minimizing the loss of useful seismic information.

When integrating a lossy compressor into the F3DT-SI workflow in an online fashion, we face new challenges. (1) Our wave-propagation simulation code has been carefully optimized and is highly efficient (Cui et al., 2009). For the California statewide simulation, evolving the entire 9-component wavefield (i.e., 3 particle-velocity components and 6 stress components, about 6.75 GB data in memory) by one time step takes around 0.163 s of wall-time on 256 cores, which amounts to about 165.64 MB/s/core of calculation speed. It is desirable if the speed of the compressor is comparable to the calculation speed. (2) The memory size per core is usually too small to allow buffering the entire time history of the simulated wavefields, which means that the online compressor has to work with the 3D spatial wavefields at only one or a few buffered time steps. This limitation prevents the online compressor from fully exploiting the data redundancy in the time dimension, which can be significant for simulated wavefields. Therefore it is harder for an online compressor to reach compression ratios comparable to those obtained by off-line compressors that can work with the full 4D wavefields. (3) Realistic F3DT-SI applications usually involve a large number of data sensitivity kernels. At the current stage of the California statewide F3DT, each F3DT-SI iteration requires ~24,000 kernel runs to construct 893,911 kernels. Each kernel run requires reading two 4D strain fields from disk repeatedly. The kernel code is heavily I/O bound. A typical kernel run spends over 90% of time on reading the two 4D strain fields repeatedly. If the compressed 4D strain fields are sufficiently small and the decompressor is sufficiently fast, we can then read the two compressed 4D strain fields once and replace slow disk reads with decompression operations to reduce the I/O overhead.

The *zfp* open source compressor (Lindstrom, 2014) adopted in this study was originally inspired by ideas from fixed-rate texture compression algorithms widely implemented in graphics hardware (e.g., Iourcha et al., 1999; Fenney, 2003; Ström and Akenine-Möller, 2005; Nystad et al., 2012), but specialized for high-precision floating-point data. We used *zfp* version 0.4.1, which improves upon the original version (Lindstrom, 2014) in both speed and compression ratio, and also supports fixed-precision and fixed-accuracy modes for setting relative respectively absolute error tolerances.

The wave-propagation simulation code is the parallel Anelastic-Wave-Propagation (*AWP*) code, which solves the 3D (visco) elastic wave-equation in the velocity-stress form using the 4th-order staggered-grid finite-difference method (Olsen, 1994; Cui et al., 2009). At each saved time step we obtain the strain fields from the stress fields using the constitutive relation (Zhao et al., 2005; 2006).

Each component of the strain tensor is represented as a 3D array of floating-point numbers inside the memory. Following most texture compression formats, as well as image and video compression formats like JPEG and MPEG, we divide the 3D array into blocks of dimensions $4 \times 4 \times 4$. (Blocks near array boundaries are padded if the array dimensions are not multiples of four.) The following 6 steps are then conceptually applied to each block independently:

1. Block-floating-point transform,
2. De-correlating transform,
3. Coefficient reordering,
4. Conversion from signed to unsigned integer,
5. Embedded coding of integers,
6. Bit stream truncation.

The truncated bit streams for all blocks of the 6 independent strain components are then aggregated and written to disk at the end of each saved time step. Steps 1, 2 and 6 are potentially lossy, while the others are guaranteed lossless. Each step is described in more detail below.

3.1. Block-floating-point transform

In this step, all the sixty-four 32-bit floating-point values in a block are normalized by expressing them with respect to the largest floating-point exponent $\hat{e}$ in the block. For efficiency of subsequent operations, each floating-point value $x = (-1)^s 2^e (1+m)$, with mantissa $0 \le m < 1$, is rewritten as a 31-bit signed integer y times some power of two, i.e., $y = \text{round}(2^{29-\hat{e}}x)$, such that $-2^{30} < y < 2^{30}$. A single-precision floating-point number has 24 bits of mantissa (including the implicit leading one) plus one sign bit. By directly transferring the 24-bit mantissa plus the sign bit to the 31-bit signed integer, we actually have gained as many as 6 bits of precision in this step. However, if the ratio of the largest value to the smallest value within a block exceeds $2^6 = 64$, then some loss of least significant bits is possible in the smaller-magnitude number. The same type of loss also happens in floating-point hardware during floating-point calculations, such as those in the finite-difference simulations. This type of loss is inherent to the floating-point format and its machine representation and can usually be ignored in practical seismic modeling. Before moving on to the next step, *zfp* transmits the 8-bit common exponent $\hat{e}$ verbatim.

3.2. De-correlating transform

The signed integers obtained from step 1 are transformed with respect to a set of basis functions to reduce their correlation. For suitably chosen basis functions, this transform often results in many near-zero coefficients, which are particularly amenable to compression. For regularly gridded 3D data, such as a component of the strain field computed in *AWP*, we adopt a separable 3D transform that can be implemented as three 1D transforms along each spatial axis. The 3D separable basis function can then be expressed as

$$B_{ijk}(x, y, z) = b_i(x) b_j(y) b_k(z) \tag{4}$$

where $0 \le i, j, k \le 3$ and $\|B_{ijk}\| = 1$. The indices i, j, k correspond to the polynomial degrees of the 1D basis functions $b_i(x)$, $b_j(y)$ and $b_k(z)$. In this study we have adopted the orthogonal Gram polynomial basis functions and an efficient lifted implementation of the 1D transform, similar to (Lindstrom, 2014). An important advantage of using those basis functions is that the matrix-vector multiplications involved in the 1D transforms can be implemented very efficiently through in-place bit-shift and addition operations. The complete 3D transform based on this *lifting scheme* involves 4.5 bit shifts and 7.5 additions (amortized) per compressed value, which compares very favorably with the 64 multiplications and 63 additions per compressed value for implementations based on conventional matrix-vector multiplications. This integer transform preserves the range of $(-2^{30}, 2^{30})$, but involves a number of divisions by two (implemented as bit shifts) that can potentially cause loss of the least significant bit. But since we have gained as many as 6 bits from the previous step, this potential loss is usually not a problem.

3.3. Coefficient reordering

The indices i, j, k of the basis functions in Eq. (4) can be considered as proxies for the wavenumber (i.e., spatial frequency) of the strain field. We use the summation of the three indices $0 \le i + j + k \le 9$ to represent the total degree of the 3D basis function. The corresponding coefficients resulting from the decorrelating transform in step 2 are then reordered according to the total degree of the 3D basis function, much akin to how 2D JPEG DCT coefficients are arranged in $i + j$ zig-zag order. The motivation for this reordering is based on the observation that the power of the strain field tends to decrease with increasing wavenumber. Thus by reordering the coefficients according to the total degree, we obtain a list of coefficients sorted roughly by their magnitude.

3.4. Conversion to unsigned integer

Before encoding the integer coefficients, we first transform them from two's complement signed integers to an unsigned representation. We have chosen the negabinary representation with base -2, for three reasons: (1) This mapping can be computed very efficiently using only one integer addition and one bitwise exclusive or. (2) Compared to other representations, it results in improved rate distortion when the bit stream is truncated, i.e., when zeroing some of the least significant bits during quantization. (3) The lack of a dedicated sign bit reduces conditionals and simplifies the logic of the encoder. Bit k in negabinary has place value $(-2)^k$, and the sign thus alternates between consecutive bits. This lossless transformation shifts the range of values from $(-2^{30}, 2^{30})$ to $[0, 7/8 \times 2^{32})$.

3.5. Embedded coding

The reordered, unsigned coefficients of a block can be transformed into a sequence of bits (i.e., a bit stream) that can be easily transmitted or stored. Several embedded coding strategies exist, and the one adopted in this study is a more efficient variation on the one documented in (Lindstrom, 2014). Because the de-correlating transform is orthogonal, all the coefficient bits in the same "bit plane" (i.e., a set of coefficient bits at the same bit position) have the same impact on L_2 error, and are thus equally "important." We therefore encode one bit plane at a time, from most to least significant bit, using a *group testing* procedure (Hong and Ladner, 2002). Anticipating that the coefficients are arranged roughly in decreasing order of magnitude, and with most of them having many leading zeros, we achieve (lossless) compression by identifying large groups of zero-bits. We begin by testing if all bits within a bit plane are zero (a negative group test), and if so emit a zero-bit and move on to the next bit plane. Otherwise (a positive group test), we output a one-bit followed by a string of coefficient value bits, in order of increasing polynomial degree, until we encounter and emit a one-bit. We then perform another group test on the remaining bits and repeat until either all remaining bits in this bit plane are zero or none remain. In this manner, the first $0 \le n \le 64$ value bits of the current bit plane are explicitly output; the remaining $64 - n$ zero-bits are encoded with a single bit. To encode the next bit plane, we first emit its first n bits, which are presumed random, before we proceed with group testing the remaining bits, incrementing n as necessary from one bit plane to the next. This coding strategy losslessly encodes all 64×32 integer coefficient bits, and usually using far fewer bits.

3.6. Bit stream truncation

By initializing all coefficients to zero on the decoder side, we note that any prefix of the resulting bit stream allows the decoder to produce a valid reconstruction of some number of leading bits of each coefficient, which after the inverse transformation results in an approximate representation of the original floating-point values. Such truncation of the bit stream affords great flexibility and granularity in choosing the compression ratio or quality. In *zfp*'s *fixed-rate* mode, the bit stream for each block is truncated at a fixed number of bits to ensure fixed-size storage for each block and random access at block granularity. (In practice, bit stream truncation is implemented by pre-empting the encoder once the bit budget has been reached). Alternative truncation strategies include *fixed precision*—by encoding a fixed number of bit planes—and *fixed accuracy*—by encoding all bit planes up to some given bit plane number q, which is equivalent to applying uniform quantization with step size 2^q. Note that in fixed-accuracy mode, the number of encoded bit planes depends both on q and on the block exponent $\hat{e}$. For a given q, a worst-case analysis reveals that the maximum reconstruction error in 3D is at most 2^{q+6}. We may, thus, choose q so as to guarantee that the decompressed data falls within a user-specified error tolerance. In this study, we used this fixed-accuracy mode of *zfp*.

4. Results

We have successfully applied our lossy online compression algorithm to significantly reduce the disk storage in the current California statewide F3DT-SI inversions. At the current stage we have carried out 4 iterations following the tomographic navigation approach as discussed in Lee et al. (2014a).

Fig. 1 shows the S-wave velocity at 10 km depth in our starting model CAF3D00 and the model CAF3D04 after the 4th iteration. About 24,000 seismic waveforms have been used in our inversion. The misfit function used in the 4 iterations was defined as the L2-norm of the frequency-dependent group-delay anomalies measured at frequencies ranging from 0.03 Hz to 0.1 Hz. Fig. 2 shows about 40% misfit reduction of about 893,000 measurements after the 4 iterations.

Synthetic waveforms computed using CAF3D04 show substantially better fit to observed waveforms than those computed using CAF3D00. We measured the differences between an observed waveform $\bar{u}(t)$ and its corresponding synthetic waveform $u(t)$ within a time window $[t_0, t_1]$ using the relative waveform misfit (RWM) defined by (e.g., Zhu and Helmberger, 1996; Tape et al., 2010; Lee et al. 2014b)

$$RWM = \frac{\int_{t_0}^{t_1} \left[\bar{u}(t) - u(t)\right]^2 dt}{\sqrt{\int_{t_0}^{t_1} \left[\bar{u}(t)\right]^2 dt \int_{t_0}^{t_1} [u(t)]^2 dt}} \tag{5}$$

Fig. 3 shows the histograms of the RWMs for CAF3D00 and CAF3D04.

Through the 4 iterations, the median of the RWM (mRWM) has dropped from 1.4 to 0.73 (Fig. 4) and the median absolute deviation (MAD) (Hoaglin et al., 1983) has dropped from 0.65 to 0.35. The mRWM and the MAD provide robust measures of the location and the spread of the RWM distributions (Lee et al., 2014b).

The inversion is being carried out on the IBM System X cluster, "Mount Moran" (284 nodes, each with two 8-core Intel E5-2670), at University of Wyoming. For this study, each *AWP* wave-propagation simulation runs on 256 CPU cores for ~35 min. Each F3DT-SI iteration requires 227 *AWP* simulations (~33,900 core-hours). Each kernel calculation takes ~1.2 min on 256 cores. We have ~24,000 kernel calculations per iteration (~122,880 core-hours). The total amount of core-hours for one F3DT-SI iteration is roughly 160,000.

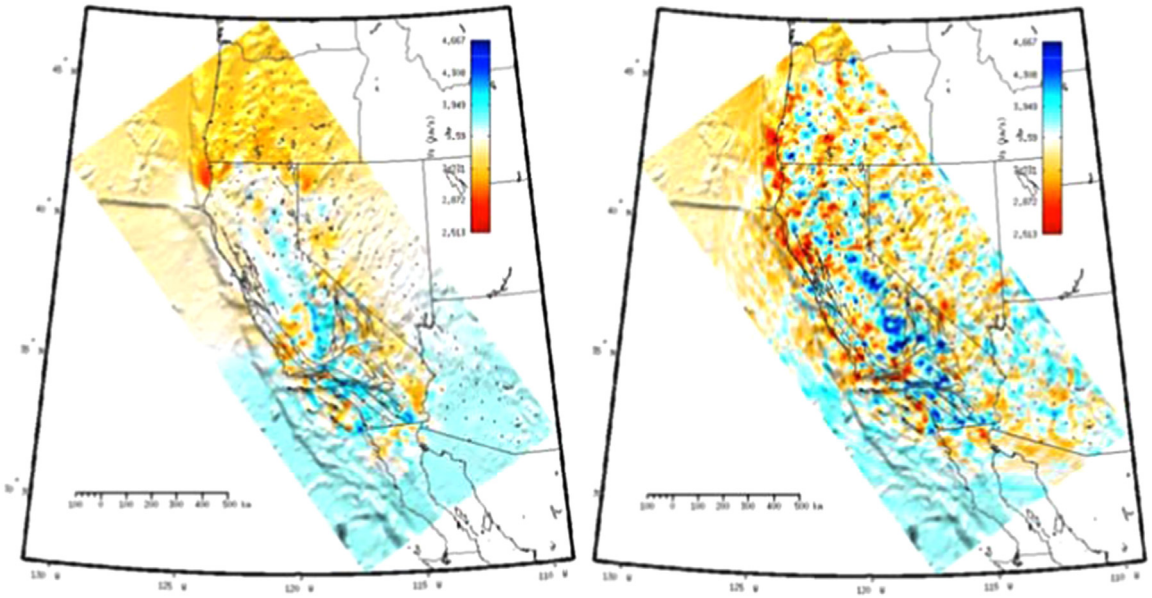

Fig. 1. Seismic shear-wave velocity (Vs) at 10 km depth in the starting model CAF3D00 (left) and in the latest updated model CAF3D04. Warm colors: lower Vs; cold colors: higher Vs. Coast lines, state boundaries and major faults are shown in black solid lines. Black dots: locations of the 227 broadband seismic stations used in this study.

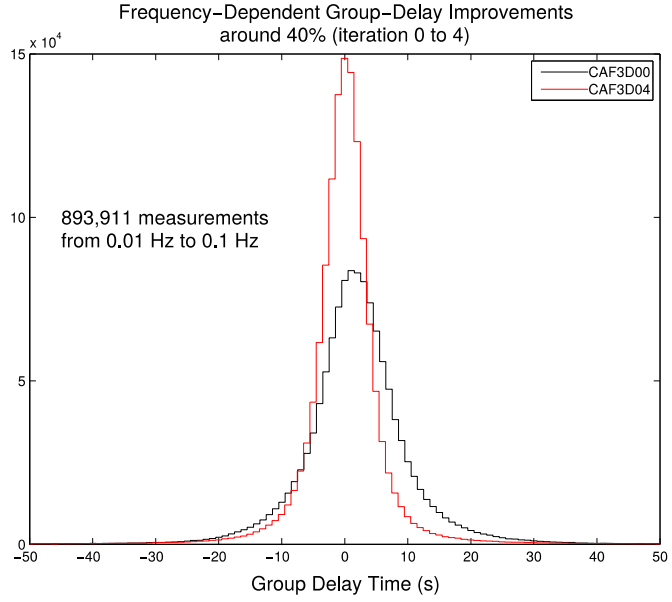

Fig. 2. Histograms show the distributions of the frequency-dependent group-delay time misfits for the starting model CAF3D00 (black) and the updated model CAF3D04 (red). The misfits quantify the discrepancies between synthetic seismograms computed using the CAF3D00 or CAF3D04 and the corresponding observed seismograms. Smaller misfits indicate better agreement between synthetic and observed seismograms. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

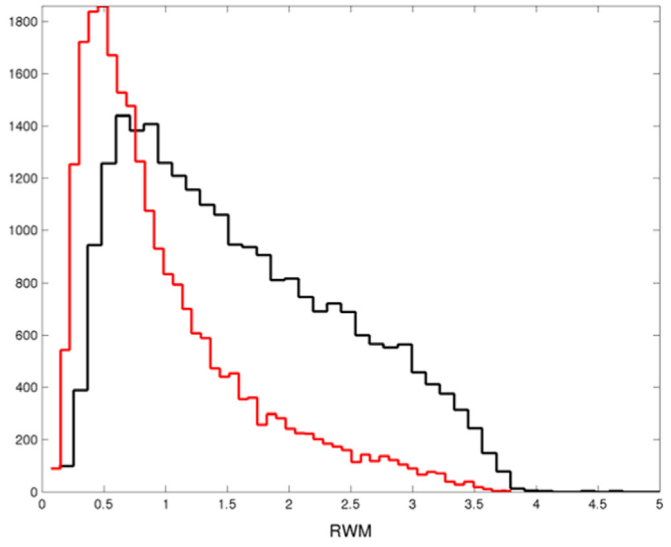

Fig. 3. Histograms show the distribution of the RWM for the starting model CAF3D00 (black) and the updated model CAF3D04 (red). The RWM quantifies the energy of the waveform differences between observed and synthetic seismograms normalized by the geometrical mean of the observed and synthetic waveform energy. Smaller RWM indicates better waveform fit. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Disk storage without compression

The *AWP* simulations are carried out on a uniform mesh with ~201.3 million grid points with 1-km grid spacing. The simulation volume is 1000-km wide, 2000-km long and 100-km deep (Fig. 1). Each *AWP* simulation computes the strain and particle-velocity fields for 12,282 times steps (~700 s). The synthetic wavefields computed using this spatial-temporal discretization set-up is accurate for frequencies up to 0.12 Hz. We are not considering viscoelastic attenuation. At the current stage, we are using the vertical components of 227 stations from the California Integrated Seismic Network (CISN) and the Transportable Array (TA) of the EarthScope project.

Along the time axis, we decimate by a factor of 10, which gives ~18 samples per dominant period of the simulated fields. Table 1 lists the disk storage with and without compression for all RSGTs with 6 different spatial decimation settings: (a) $D_x=D_y=8$, $D_z=2$; (b) $D_x=D_z=8$, $D_y=2$; (c) $D_x=D_y=4$, $D_z=2$; (d) $D_x=D_z=4$, $D_y=2$; (e) $D_x=D_y=D_z=2$; (f) no decimation. Here $D_{x,y,z}$ are decimation rates in x, y and z, respectively. At each saved grid point and time step, we store the 6 independent components of the strain tensor and the 3 components of the particle velocity in single-precision. At the current stage we are using setting (c).

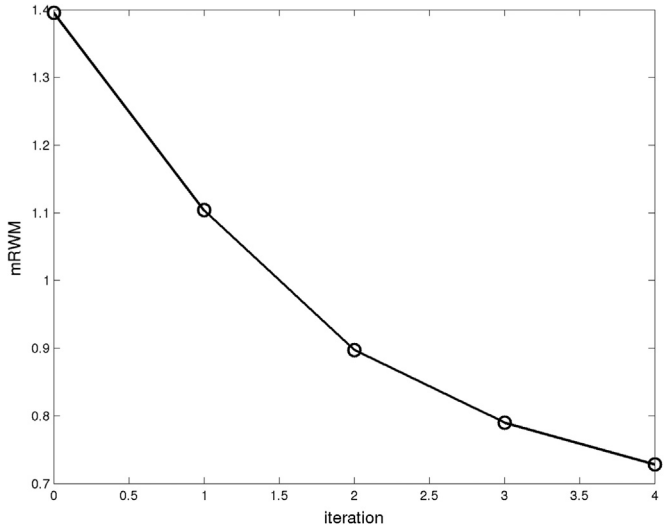

Fig. 4. The median of the RWM (mRWM) at different iteration number. At iteration 0, synthetic seismograms were computed using the starting model CAF3D00 and at iteration 4, synthetics were computed using the updated model CAF3D04.

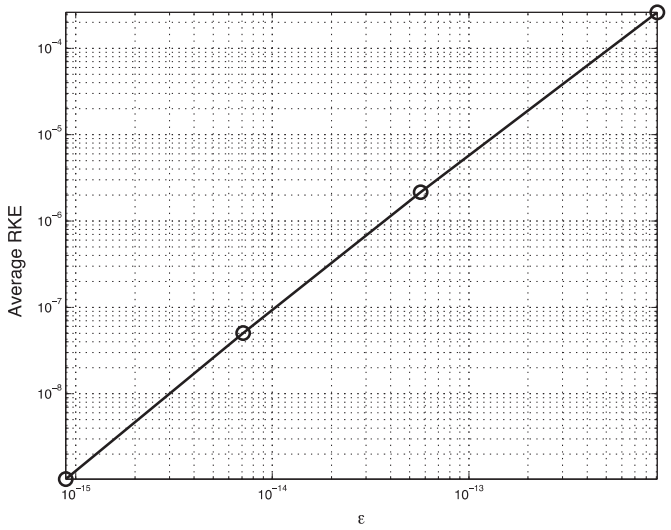

Fig. 5. The average relative kernel error (*RKE*) as defined in Eq. (6) in the text as a function of the error tolerance ε used by the compressor for truncating the bit streams. A larger error tolerance indicates that more bits in the bit stream are being discarded. Larger average *RKE* indicates larger discrepancies between the kernels computed using the uncompressed strain fields and those computed using the compressed strain fields.

4.2. Error tolerance

The average compression ratio mainly depends upon two factors: (1) the criteria used for truncating the bit stream (Sections 3.6) and (2) the spatial smoothness (i.e., redundancy) of the wavefields. When truncating the bit stream, we used an error tolerance of $\varepsilon = 2^t$. All coefficient bits lying on a bit plane below 2^{t-6} are discarded. The effective precision in terms of the number of bits used for representing each integer coefficient is $p = (32 - 29) + \hat{e} - (t - 6) = \hat{e} - t + 9$ restricted to $0 \le p \le 32$, where $\hat{e}$ is the exponent associated with the block.

In practice, we determine the error tolerance ε through trial-and-error based on the relative kernel error (*RKE*) defined as

$$RKE = \frac{\int_V \mathrm{d}V(\mathbf{x})\left[\bar{K}(\mathbf{x}) - K(\mathbf{x})\right]^2}{\sqrt{\int_V \mathrm{d}V(\mathbf{x})\bar{K}^2(\mathbf{x}) \int_V \mathrm{d}V(\mathbf{x})K^2(\mathbf{x})}} \tag{6}$$

Here $K(\mathbf{x})$ and $\bar{K}(\mathbf{x})$ are the data sensitivity kernels computed using the strain fields with and without compression, respectively. Fig. 5 shows the average *RKE* (averaged over 1000 kernels with respect to the two Lamé parameters) as a function of ε. We computed the average *RKE* using 4 different error tolerances: $\varepsilon = 2^{-40} \approx 9.09 \times 10^{-13}$, $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$, $\varepsilon = 2^{-47} \approx 7.11 \times 10^{-15}$, $\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$. For the strain fields used in our inversion, when $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$, the average *RKE* is $\sim 10^{-6}$ (Fig. 5). Examples of the frequency-dependent group-delay kernels with respect to the Lamé parameter μ computed using the strain fields without compression and those computed using the strain fields compressed with the 4 different ε values are shown in Fig. 6. The compression ratios in Table 1 were computed using $\varepsilon = 2^{-44}$. Compression-introduced kernel errors have negligible effects on the inverted structure model for two reasons: (1) the relative kernel errors (Fig. 5) are much smaller than the estimated noise level in the misfit measurements (a few percent for the dataset used in our current inversion); (2) compression-introduced errors are random and tend to cancel each other's effects during the LSQR inversion step (Zhao et al., 2005; 2006).

Even though *zfp* is a block-based compression algorithm, it is more resilient to block artifacts than traditional block-based compression algorithms, such as JPEG. For the error tolerances we have tried, no block artifacts were observed in the compressed strain fields or the kernels. Fig. 7 shows a snapshot of the *zz*-component of the strain tensor at 2-km depth with no compression and those compressed using the 4 different error tolerances. This lack of block artifacts is due to (1) the smaller block size ($4 \times 4 \times 4$ in *zfp* compared with 8×8 in JPEG), which reduces the mismatch across block boundaries; (2) the fixed-accuracy mode in *zfp*, which bounds the error across blocks to a uniform value; and (3) *zfp*'s uniform quantization of transform coefficients by bit plane (Lindstrom 2014).

4.3. Compression ratios

Compression ratios for the 6 grid decimation settings are summarized in Table 1 and Fig. 8. At lower spatial decimation rates

Table 1
RSGT disk storage with and without compression for different decimation settings. D_x, D_y and D_z are decimation rates in the two horizontal axes *x*, *y* and the vertical axis *z*, respectively.

Spatial decimation rates			Number of saved grid points	Saved time steps	Number of RSGTs	Disk storage without compression (TB)	Disk storage with compression (TB)	Average compression ratio
D_x	D_y	D_z						
8	8	2	1,572,864	1228	227	14.36	0.98	14.69
8	2	8	1,572,864	1228	227	14.36	0.99	14.56
4	4	2	6,291,456	1228	227	57.42	2.89	20.03
4	2	4	6,291,456	1228	227	57.42	2.91	19.83
2	2	2	25,165,824	1228	227	229.69	8.95	25.86
1	1	1	201,326,592	1228	227	1837.50	48.09	38.37

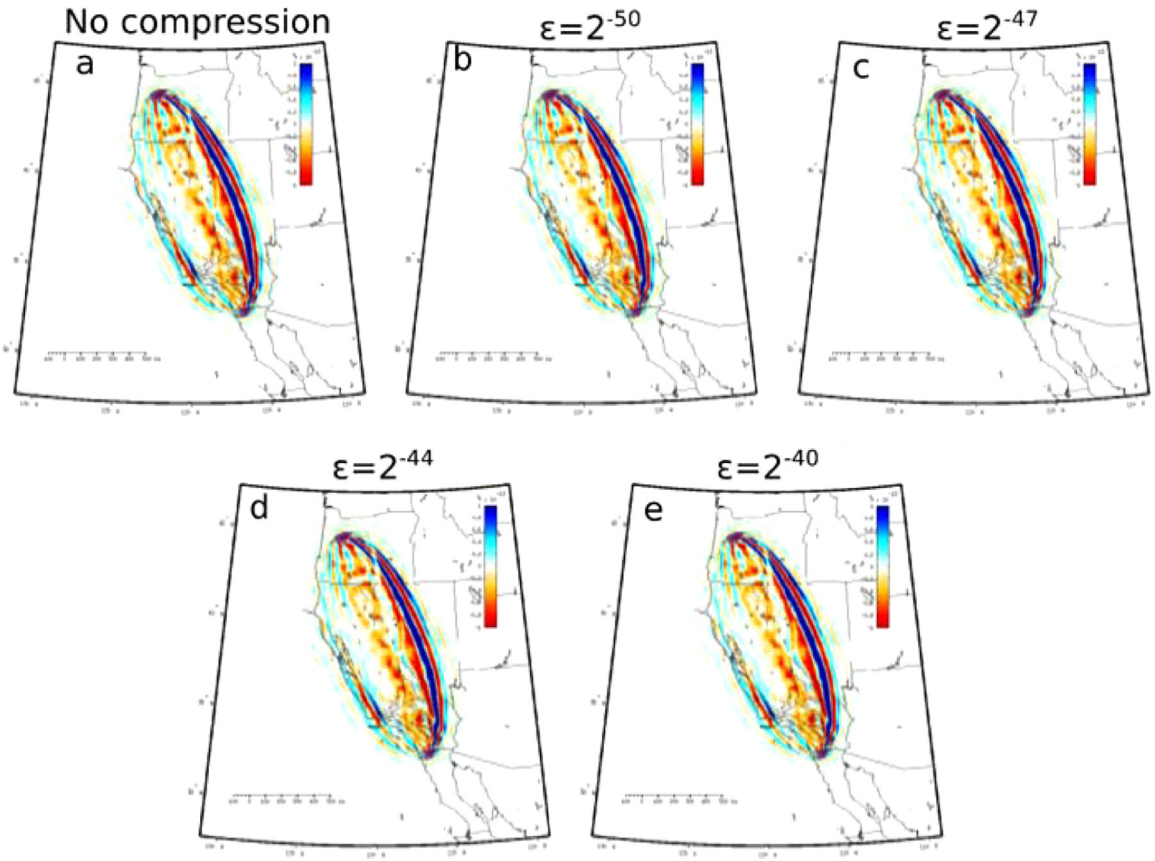

Fig. 6. An example of the data sensitivity kernels computed using raw strain fields without compression (a) and those computed from strain fields compressed at error tolerance $\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$ (b), $\varepsilon = 2^{-47} \approx 7.11 \times 10^{-15}$ (c), $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$ (d) and $\varepsilon = 2^{-40} \approx 9.09 \times 10^{-13}$(e). Map-view plots of the data sensitivity kernels are at 10 km depth. Warm colors: negative sensitivity; cold colors: positive sensitivity. The kernel is computed for a frequency-dependent group-delay misfit measured at 0.1 Hz on a vertical-component Rayleigh wave with respect to the second Lamé parameter μ.

the correlation among the strain values on neighboring saved grid points is higher and the de-correlating transform allows us to compress more redundant information efficiently, which increases the average compression ratio. As shown in Table 1 and Fig. 8, the average compression ratio increases from about 15 for the highest decimation rate to about 38 (i.e., less than one bit per value) for the lowest decimation rate.

The total number of saved grid points for decimation setting (a) is identical to that in setting (b). In setting (a), the decimation rate is 2 on the z-axis and 8 on the y-axis, while in setting (b) the decimation rate is 8 on the z-axis and 2 on the y-axis. The average compression ratios for these two settings are similar, indicating that on average the redundancy (smoothness) of the strain fields in the horizontal dimensions is similar to that in the vertical dimension. For decimation settings (c) and (d), we observe similar results.

4.4. Compression performance in wave-propagation simulations

The *AWP* simulation code was parallelized using the message-passing interface (MPI) and a domain decomposition approach. The uniform mesh used in the finite-difference simulations was divided evenly into a number of sub-meshes and each CPU core handles the computation, compression and output of the wavefields on one sub-mesh.

During the simulation we measured the time used for compressing the wavefields and for writing the compressed fields to disk on each core. The wall time for each run was obtained by taking the maximum value of the estimates generated by all the CPU cores.

Fig. 9 shows the performance improvements for the *AWP* simulations using our online compressor with 4 different error tolerances. The amount of time used for compression combined with writing the compressed wavefields to disk is less than the amount of time used for writing the uncompressed wavefields to disk. The compression time increases with decreasing error tolerance. For smaller error tolerance, more bit planes need to be encoded, therefore more time is spent on compression. For $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$, the compression time was ~8.87 s and the time spent on writing the compressed wavefields to disk was ~2.36 s. The compression-output combined time, 11.23 s, is about one fourth of the time spent on writing the raw wavefields to disk (Fig. 9). The average speed of the online compressor was ~116.80 MB/s/core. For the higher error tolerance $\varepsilon = 2^{-40} \approx 9.09 \times 10^{-13}$, the number of bit planes need to be encoded is fewer and the speed of the compressor was ~137.35 MB/s/core. For the lower error tolerance $\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$, the compression speed was ~87.86 MB/s/core.

4.5. Decompression performance in kernel calculations

The kernel calculation involves temporal convolutions of two strain fields at every saved grid point. The kernel code, named *ker*, is parallelized using a domain decomposition approach, in which each MPI process computes the kernel on the saved grid points of one sub-mesh. The number of sub-meshes (i.e., MPI processes) is determined by the *AWP* simulation. If each process has access to sufficient amount of memory, only two disk-read operations per process are needed to load the two 4D strain fields on one sub-mesh.

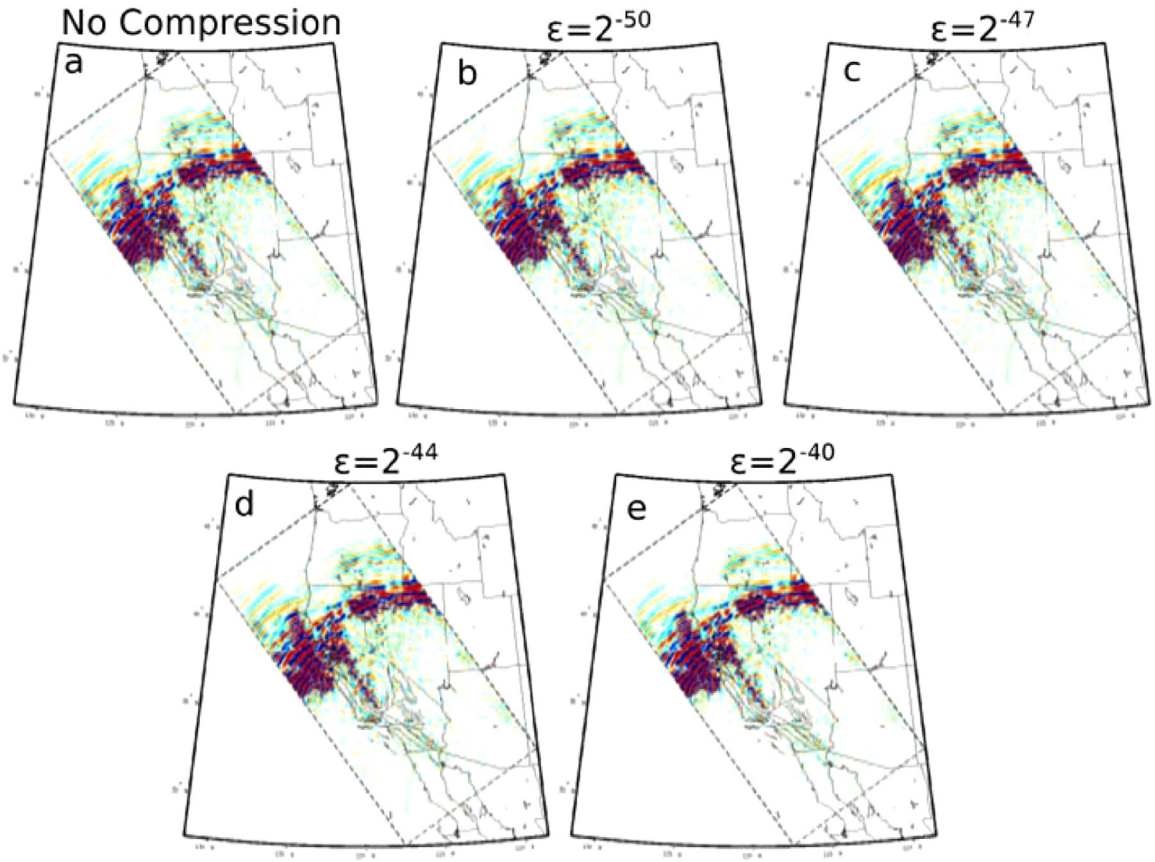

Fig. 7. Comparison of the snapshots of 4D *zz*-component strain fields at 2-km depth without compression (a), compressed at error tolerance $\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$ (b), error tolerance $\varepsilon = 2^{-47} \approx 7.11 \times 10^{-15}$ (c), error tolerance $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$ (d) and error tolerance $\varepsilon = 2^{-40} \approx 9.09 \times 10^{-13}$ (e).

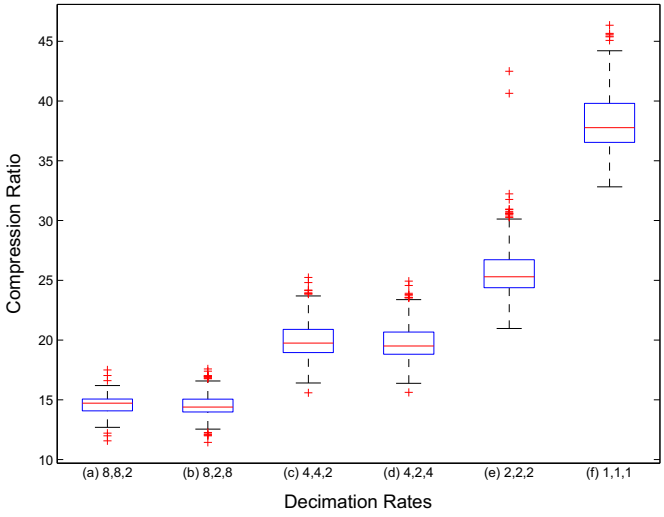

Fig. 8. Box plot of the RSGT compression ratios for the 6 different grid decimation settings discussed in the text at the error tolerance $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$. The red line inside each box indicates the median over the 227 RSGTs. The top and bottom edges of each box indicate the 75th and 25th percentile, respectively. The whiskers extend to approximately ± 2.7 standard deviations and points lying outsize of the ranges are plotted as red plus symbols. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

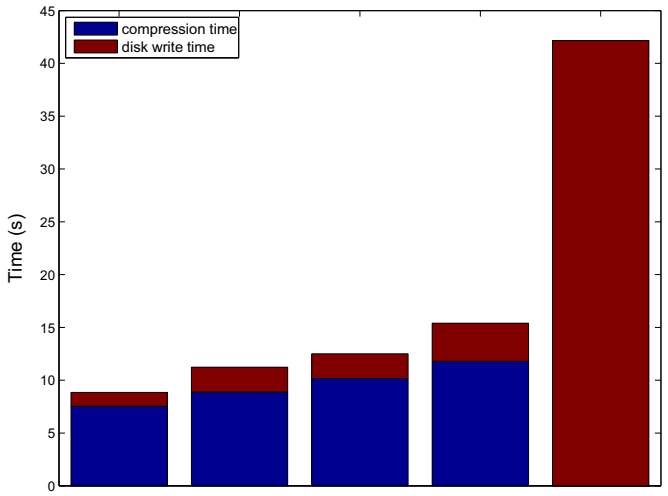

Fig. 9. The average time for compressing and writing the strain fields in *AWP* wave-propagation simulations for the spatial decimation setting (c). From left to right, the columns are for error tolerance values $\varepsilon = 2^{-40} \approx 9.09 \times 10^{-13}$, $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$, $\varepsilon = 2^{-47} \approx 7.11 \times 10^{-15}$, $\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$ and no compression. Blue: compression time; red: disk-writing time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The memory-per-core on Mount Moran (~2GB), and also many other small- to medium-sized clusters, is often not sufficient to hold the entire time history of two 4D strain fields. In our inversion, the 4D wavefields on all saved grid points (decimation setting c) and all saved time steps of one sub-mesh occupies about 1.1 GB without compression. To hold two 4D strain fields on one sub-mesh, we would need 2.2 GB/core, not including memory for other data.

The approach adopted in *ker* is to divide each sub-mesh into smaller portions so that the entire time history of the strain fields on each portion can fit inside the memory of each process. For each disk-read, we load the entire time history of the strain fields

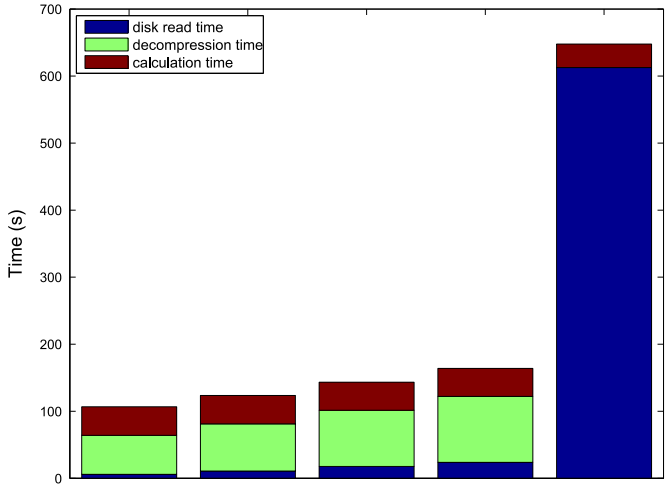

Fig. 10. The average time for reading the compressed strain fields from disk (blue), decompression (green) and kernel computation (red) in the new kernel code *kerz* and the original kernel code *ker*. From left to right, the columns are for error tolerance values $\varepsilon = 2^{-40} \approx 9.09 \times 10^{-13}$, $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$, $\varepsilon = 2^{-47} \approx 7.11 \times 10^{-15}$, $\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$ and no compression. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

on one portion of a sub-mesh, compute the kernels on that portion and then repeat on every portion. The number of disk-reads depends upon the memory size per core. The code *ker* worked well on supercomputers with powerful I/O sub-systems but its performance deteriorates significantly on smaller clusters.

The compressed strain fields have much smaller sizes than the raw strain fields. For the California statewide F3DT, even at the smallest error tolerance ($\varepsilon = 2^{-50} \approx 8.88 \times 10^{-16}$), the maximum size of the 4D strain field on a sub-mesh is about 364 MB. It is now feasible to load the entire time history of two compressed 4D strain fields of one sub-mesh into the memory of one core, which takes two disk-reads per process. We still divide each sub-mesh into a number of smaller portions, whose sizes depend upon the remaining memory. We decompress the 4D strain fields on one portion of the sub-mesh, compute the kernels on that portion and repeat on every portion. We therefore replace multiple disk-reads for loading the strain fields of all the portions with the same number of decompression operations. The speed of the decompressor is therefore critical for the overall performance of our new kernel code *kerz*.

Fig. 10 shows the performance of *kerz* compared with the performance of *ker* for the 4 compression error tolerances. For $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$ used in our inversion, the speed of *kerz* is about 5.3 times that of *ker*. The amount of time spent on computing the kernels is roughly the same in *kerz* at the 4 different error tolerance values and also in *ker*. The amount of time spent on reading the compressed 4D wavefields from disk increases slightly with decreasing error tolerance due to the fact that the size of the compressed wavefields increases with decreasing error tolerance. The amount of time used for decompressing the two 4D wavefields also increases with decreasing error tolerance. For $\varepsilon = 2^{-44} \approx 5.68 \times 10^{-14}$ the decompression speed is ~353.52 MB/s/core.

5. Summary and discussion

In this study, we have shown that by integrating lossy compression into the F3DT-SI workflow, we can substantially reduce disk storage and improve I/O performance, which makes it feasible to adopt F3DT-SI to solve practical F3DT problems of realistic size on small clusters. The main idea of our implementation was to utilize the compute cycles previously wasted in waiting for the I/O of strain fields to compress/decompress the strain fields. Because the online compressor can be implemented very efficiently, we not only reduced disk storage substantially but also improved the overall I/O performance of the entire F3DT-SI workflow significantly.

The effectiveness of our compressor can be further enhanced. At the current stage, only the redundancies in the 3 spatial dimensions have been exploited by the compressor. It is also possible to reduce the redundancy in the temporal dimension by using a 4D separable basis function in the de-correlating transform.

Even though the compressor can effectively reduce the redundancy within the wavefields, it does not account for smoothing during kernel calculations and through smoothness damping when solving the linear system in Eq. (3). The redundancy introduced through those smoothing processes can be reduced more effectively through a judicious choice of the spatial decimation rates. In practice, combining spatial decimation with online compression can provide higher overall storage reduction without compromising the accuracy of the inverted structure model.

Acknowledgment

The work performed at University of Wyoming was supported by the United States Geological Survey under Grant number G10AP00032 and Crust LLC. This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This material is based upon work supported by the U. S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research.

References

Al-Moohimeed, M.A., 2004. Towards an efficient compression algorithm for seismic data. In: Proceedings of IEEE Radio Science Conference, 2004. Asia-Pacific, pp. 550–553.

Anderson, D.L., Dziewonski, A.M., 1984. Seismic tomography. Sci. Am. 251, 60–68.

Aparna, P., David, D.S., 2006. Adaptive local cosine transform for seismic image compression. In: Proceedings of IEEE International Conference on Advanced Computing and Communications. ADCOM 2006, pp. 254–257.

Aqrawi, A., Elster, A.C., 2011. Bandwidth reduction through multithreaded compression of seismic images. In: Proceedings of 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), pp. 1730–1739.

Averbuch, A.Z., Meyer, F., Strömberg, J.O., Coifman, R., Vassiliou, A., 2001. Low bit-rate efficient compression for seismic data. IEEE Trans. Image Process. 10 (12), 1801–1814.

Chen, P., Jordan, T.H., Zhao, L., 2005. Finite-moment tensor of the 3 September 2002 Yorba Linda earthquake. Bull. Seism. Soc. Am. 95 (3), 1170–1180.

Chen, P., Zhao, L., Jordan, T.H., 2007a. Full 3D tomography for the crustal structure of the Los Angeles region. Bull. Seism. Soc. Am. 97 (4), 1094–1120.

Chen, P., Jordan, T.H., Zhao, L., 2007b. Full three-dimensional tomography: a comparison between the scattering-integral and adjoint-wavefield methods. Geophys. Journal. Int. 170 (1), 175–181.

Chen, P., Lee, E.J., 2015. Full-3D Seismic Waveform Inversion: Theory, Software and Practice. Springer, Berlin, Germany, p. 511.

Cui, Y., Olsen, K., Chourasia, A., Moore, R., Maechling, P., Jordan, T., 2009. The TeraShake computational platform for large-scale earthquake simulations. In: Xing, H. (Ed.), Advances in Geocomputing. Springer, Berlin, Heidelberg, pp. 229–277.

Fajardo, C., M Reyes, O., Ramirez, A., 2015. Seismic data compression using 2D lifting-wavelet algorithms. Ing. Y. Cienc. 11 (21), 221–238.

Fenney, S., 2003. Texture compression using low-frequency signal modulation. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. Eurographics Association, pp. 84–91.

Fichtner, A., Kennett, B.L., Igel, H., Bunge, H.P., 2009. Full seismic waveform tomography for upper-mantle structure in the Australasian region using adjoint methods. Geophys. J. Int. 179 (3), 1703–1725.

Hoaglin, D.C., Mosteller, F., Tukey, J.W., 1983. Understanding Robust and Exploratory Data Analysis. John Wiley & Sons, United States, pp. 404–414, ISBN:

0-471-09777-2.
Hong, E.S., Ladner, R.E., 2002. Group testing for image compression. IEEE Trans. Image Process. 11 (8), 901–911.
Iourcha, K.I., Nayak, K.S., Hong, Z., 1999. U.S. Patent No. 5,956,431. U.S. Patent and Trademark Office, Washington, DC.
Iyer, H.M., 1993. Seismic Tomography: Theory and Practice. Springer Science & Business Media, Berlin, Germany.
Jonsson, S.B., Spanias, A.S., 1990. Seismic data compression. In: Proceedings of the IEEE Ninth Annual International Phoenix Conference on Computers and Communications, 1990. Conference, pp. 276–279.
Lee, E.J., Chen, P., Jordan, T.H., Maechling, P.B., Denolle, M.A., Beroza, G.C., 2014a. Full-3-D tomography for crustal structure in Southern California based on the scattering-integral and the adjoint-wavefield methods. J. Geophys. Res.: Solid Earth 119 (8), 6421–6451.
Lee, E.J., Chen, P., Jordan, T.H., 2014b. Testing waveform predictions of 3D velocity models against two recent Los Angeles earthquakes. Seism. Res. Lett. 85 (6). http://dx.doi.org/10.1785/0220140093.
Lee, E.J., Huang, H., Dennis, J.M., Chen, P., Wang, L., 2013. An optimized parallel LSQR algorithm for seismic tomography. Comput. Geosci. 61, 184–197.
Lervik, J.M., Rosten, T., Ramstad, T., 1996. Subband seismic data compression: optimization and evaluation. In: Proceedings of the IEEE Workshop on Digital Signal Processing, pp. 65–68.
Lindstrom, P., Isenburg, M., 2006. Fast and efficient compression of floating-point data. IEEE Trans. Vis. Comput. Graph. 12 (5), 1245–1250.
Lindstrom, P., 2014. Fixed-rate compressed floating-point arrays. IEEE Trans. Vis. Comput. Graph. 20 (12), 2674–2683.
Mandyam, G., Magotra, N., McCoy, W., 1996. Lossless seismic data compression using adaptive linear prediction. In: Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing. IGARSS'96. 'Remote Sensing for a Sustainable Future', vol. 2, pp. 1029–1031).
Nolet, G., 1987a. Seismic wave propagation and seismic tomography. In: Nolet G. (Ed.), Seismic Tomography. Springer, Netherlands, pp. 1–23.
Nolet G., 1987b. Waveform tomography In: Nolet G. (Ed.), Seismic Tomography. Springer, Netherlands, pp. 301–322.
Nolet, G., 2008. A Breviary of Seismic Tomography. A Breviary of Seismic Tomography, by Guust Nolet. Cambridge University Press, Cambridge, UK, p. 1.
Nolet, G. (Ed.), 2012. Seismic Tomography: with Applications in Global Seismology and Exploration Geophysics 5. Springer Science & Business Media, Berlin, Germany.
Nystad, J., Lassen, A., Pomianowski, A., Ellis, S., Olson, T., 2012. Adaptive scalable texture compression. In: Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics. Eurographics Association, pp. 105–114.
Olsen, K.B., 1994. Simulation of Three-dimensional Wave Propagation in the Salt Lake Basin. University of Utah, Salt Lake City, UT.
Ström, J., Akenine-Möller, T., 2005. i PACKMAN: High-quality, low-complexity texture compression for mobile phones. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. ACM, pp. 63–70.
Tape, C., Liu, Q., Maggi, A., Tromp, J., 2010. Seismic tomography of the southern California crust based on spectral-element and adjoint methods. Geophys. J. Int. 180 (1), 433–462.
Vassiliou, A.A., Wickerhouser, M.V., 1997. Comparison of wavelet image coding schemes for seismic data compression. In: Optical Science, Engineering and Instrumentation'97. International Society for Optics and Photonics, pp. 118–126.
Villasenor, J.D., Ergas, R.A., Donoho, P.L., 1996. Seismic data compression using high-dimensional wavelet transforms. In: Proceedings of IEEE Conference on Data Compression DCC'96, pp. 396–405.
Wang, X.Z., Teng, Y.T., Gao, M.T., Jiang, H., 2004. Seismic data compression based on integer wavelet transform. Acta Seism. Sin. 17 (1), 123–128.
Wang, Y., Wu, R.S., 2000. Seismic data compression by an adaptive local cosine/sine transform and its effects on migration. Geophys. Prospect. 48 (6), 1009–1031.
Wood, L.C., 1974. Seismic data compression methods. Geophysics 39 (4), 499–525.
Wu, W., Yang, Z., Qin, Q., Hu, F., 2006. Adaptive seismic data compression using wavelet packets. In: Proceedings of IEEE International Conference on Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. pp. 787–789.
Xie, X., Qin, Q., 2009. Fast lossless compression of seismic floating-point data. In: Proceedings of IEEE International Forum on Information Technology and Applications, 2009. IFITA'09, vol. 1, pp. 235–238.
Zhao, L., Jordan, T.H., Olsen, K.B., Chen, P., 2005. Fréchet kernels for imaging regional earth structure based on three-dimensional reference models. Bull. Seism. Soc. Am. 95 (6), 2066–2080.
Zhao, L., Chen, P., Jordan, T.H., 2006. Strain Green's tensors, reciprocity, and their applications to seismic source and structure studies. Bull. Seism. Soc. Am. 96 (5), 1753–1763.
Zheng, F., Liu, S., 2012. A fast compression algorithm for seismic data from non-cable seismographs. In: Proceedings of IEEE World Congress on Information and Communication Technologies (WICT) 2012, pp. 1215–1219.
Zhu, L., Helmberger, D.V., 1996. Advancement in source estima- tion techniques using broadband regional seismograms. Bull. Seism. Soc. Am. 86 (5), 1634–1641.