DOI: 10.14529/jsfi210201

Accelerating Seismic Redatuming Using Tile Low-Rank Approximations on NEC SX-Aurora TSUBASA

Yuxi Hong[1]***, Hatem Ltaief***[1]***, Matteo Ravasi***[1]***, Laurent Gatineau***[2]***, David Keyes***[1]

© The Authors 2021. This paper is published with open access at SuperFri.org

With the aim of imaging subsurface discontinuities, seismic data recorded at the surface of the Earth must be numerically re-positioned inside the subsurface where reflections have originated, a process referred to as redatuming. The recently developed Marchenko method is able to handle full-wavefield data including multiple arrivals. A downside of this approach is that a multi-dimensional convolution operator must be repeatedly evaluated to solve an expensive inverse problem. As such an operator applies multiple dense matrix-vector multiplications (MVM), we identify and leverage the data sparsity structure for each frequency matrix and propose to accelerate the MVM step using tile low-rank (TLR) matrix approximations. We study the TLR impact on time-to-solution for the MVM using different accuracy thresholds whilst at the same time assessing the quality of the resulting subsurface seismic wavefields and show that TLR leads to a minimal degradation in terms of signal-to-noise ratio on a 3D synthetic dataset. We mitigate the load imbalance overhead and provide performance evaluation on two distributed-memory systems. Our `MPI+OpenMP` TLR-MVM implementation reaches up to 3X performance speedup against the dense MVM counterpart from NEC scientific library on 128 NEC SX-Aurora TSUBASA cards. Thanks to the second generation of high bandwidth memory technology, it further attains up to 67X performance speedup compared to the dense MVM from Intel MKL when running on 128 dual-socket 20-core Intel Cascade Lake nodes with DDR4 memory. This corresponds to 110 TB/s of aggregated sustained bandwidth for our TLR-MVM implementation, without suffering deterioration in the quality of the reconstructed seismic wavefields.

Keywords: seismic redatuming, tile low-rank approximations, matrix-vector multiplication, load balancing, high bandwidth memory, NEC SX-Aurora TSUBASA.

Introduction

Exploration geophysics is an applied branch of geophysics that uses several physical measurements at the surface of the Earth (e.g., seismic, gravity, electromagnetic) to estimate the physical properties of the first few kilometers of the subsurface. Originally developed with the aim of mapping anomalies corresponding to mineral or hydrocarbon accumulations, these methods are nowadays also used in the context of geothermal exploration, carbon capture, and storage evaluation and monitoring, as well as to assess the integrity of the near subsurface for offshore wind farms.

Seismic reflection is a popular remote sensing technique that utilizes reflected seismic waves to produce high-resolution images of the geological structures as well as estimates of elastic properties of the subsurface. Its success is motivated by the fact that the waves propagating in the subsurface and being recorded at the surface of the Earth are governed by the well known elastic wave equation; various techniques have been developed during the years to harness the information contained in such recordings – see [35] for a detailed treatise. With the aim of imaging subsurface discontinuities, seismic data recorded at the surface of the Earth must be numerically re-positioned at locations in the subsurface where reflections have originated, a process generally referred to as *redatuming* by the geophysical community [7]. Historically, this

[1]King Abdullah University of Science and Technology, Kindom of Saudi Arabia
[2]NEC Deutschland GmbH, HPC Division

process has been carried out by numerically time-reversing the data recorded along an open boundary of surface receivers into the subsurface. Despite its simplicity, such an approach is only able to handle seismic energy from primary arrivals (i.e., waves that interact only once with the medium discontinuities) failing to explain multi-scattering in the subsurface. As a result, seismic images are contaminated by artificial reflectors if data are not pre-processed prior to imaging such that multiples are removed from the data. In the last decade, a novel family of methods has emerged under the name of *Marchenko redatuming* [12, 30, 33]. Such methods allow for accurate redatuming of the full-wavefield recorded seismic data including multiple arrivals. This is achieved by solving an inverse problem, the adjoint modeling of which can be shown to be equivalent to the standard single-scattering redatuming method of [7]. Whilst being more accurate, this new approach calls for solution of an inverse problem that requires repeated application of the so-called multi-dimensional convolution (MDC) operator and its adjoint. Mostly because of the extremely expensive nature of these operators in terms of complexity and memory footprint, the *Marchenko redatuming* method has not been widely adopted by the geophysical community yet. It also shows poor scalability due to their inherent memory-bound behavior.

In this paper, we report on accelerating these expensive operators by using Tile Low-Rank (TLR) approximations to perform one of the most time-consuming computational kernels, i.e., the Matrix-Vector Multiplication (MVM) operation, on the NEC vector computing SX-Aurora TSUBASA hardware solution. In fact, MVM is the workhorse of *Marchenko redatuming* for seismic imaging since it is repeatedly applied for hundreds of frequencies at each step of the iterative process. Instead of operating the MVM on the original dense data structure, our numerical technique consists in (1) splitting it into tiles with elements contiguously stored in memory, (2) compressing the tile matrix using TLR approximations (e.g., using randomized SVD [21]) up to an application-dependent accuracy threshold, and (3) performing the MVM directly on the compressed TLR data storage. This translates into a reduction of the number of floating-point operations, while further saving memory footprint. The latter is especially critical when working with large 3D seismic datasets. This TLR algorithmic redesign of the MVM introduces load imbalance when processing low and high frequencies that does not occur with the traditional dense MVM. Indeed, TLR matrices associated with high frequencies reveal higher ranks than those from low frequencies. We design and implement a load balancing technique to map processing units into frequencies so that the overall application's idle time is limited. Our `MPI+OpenMP` TLR-MVM implementation saturates the second generation of high bandwidth memory (HBM2) from the SX-Aurora TSUBASA cards and maintains a decent scalability when increasing the number of vector engines. We assess the accuracy of TLR-MVM and demonstrate its numerical robustness on representative 3D seismic datasets. We benchmark our TLR-MVM on two distributed-memory systems. It reaches up to 3X performance speedup against the dense MVM counterpart from NEC scientific library on 128 NEC SX-Aurora TSUBASA cards. Thanks to HBM2, it further attains up to 67X performance speedup in time compared to the dense MVM from Intel MKL (i.e., the `CGEMV` kernel) when running on 128 dual-socket 20-core Intel Cascade Lake nodes with DDR4 memory. This corresponds to 110 TB/s of aggregated sustained bandwidth for our TLR-MVM implementation.

The contributions of this paper are as follows. We democratize the *Marchenko redatuming* method for seismic imaging simulations by integrating TLR-MVM as the core computational engine for solving the inverse problem. We conduct performance profiling of our TLR-MVM code

using NEC *Ftrace* profiler tool and identify hot spots and room for improvement. In particular, we mitigate the load imbalance engendered by TLR-MVM when operating matrices from several frequencies in an embarrassingly parallel fashion. We highlight the performance advantage of TLR-MVM over the traditional dense MVM on Intel x86 and NEC vector computing hardware solution, without suffering deterioration in the image quality. We evaluate our implementation using the roofline model [34] and show how TLR-MVM is able to leverage HBM2 technology.

The remainder of the paper is as follows. Section 1 presents related work on low-rank matrix approximations. Section 2 describes the seismic *Marchenko redatuming* method. Section 3 recalls the general TLR-MVM algorithm. Section 4 provides implementation details of multiple inlined TLR-MVM calls and introduces the necessary load balancing strategies. Section 5 shows the TLR impact on numerical accuracy using proxy 3D seismic datasets. Section 6 reports on performance analysis and experimental results of TLR-MVM on two distributed-memory systems composed of x86 and vector computing architectures. Section 7 discusses potential future work along this herein research direction, and we conclude in Section 7.

1. Related Work

Low-rank matrix approximation is a class of algebraic compression methods, that permits to exploit the data sparsity of large matrices. This becomes critical when performing linear algebra operations on these large operators since the algorithmic complexity and the memory footprint can be reduced [9, 18].

While the literature is rich in the theory of low-rank approximations, e.g., hierarchical matrix ($\mathcal{H}$-matrix) low-rank format [17, 20], supporting weak [16, 31] and strong [8] admissibility, or flat tile low-rank (TLR) matrix approximations [5], there exist only a few works on HPC implementations targeting x86 [2–4, 13, 24] and GPU hardware accelerators [10, 14, 23].

For instance, in the context of wave-equation-based seismic processing methods, the estimation of primaries by sparse inversion [22] (EPSI) suggests that low-rank approximations of the integral operators may be utilized to reduce the storage and computation cost of the MVM. This work is however only applied as a proof-of-concept to 2D datasets using the Hierarchical Semi-Separable (HSS) compression data format. While HSS provides linear complexity, it may face challenges in compressing 3D datasets resulting in an increase of the arithmetic complexity. Moreover, one of the main reasons that slows down the wide adoption of low-rank approximations in scientific applications on current petascale supercomputers is the lack of support for advanced numerical kernels from the vendor numerical libraries. Indeed, $\mathcal{H}$-matrix computations require the development of new kernels that are versatile enough to effectively support a range of arithmetic intensity, while exhibiting low overheads during kernel launch. On the contrary, flat TLR matrix approximations is a pragmatic approach, which represents a compromise between algorithmic complexity and software development/deployment on emerging HPC platforms.

Referred as batched matrix operations [1, 10, 15], the idea behind these advanced numerical kernels is to simultaneously execute many linear algebra kernels accessing different matrices so that one may achieve high hardware occupancy. While the support from optimized vendor libraries has improved over the last few years, developers may still have to implement their own kernels (e.g., on GPUs) or simply fall back to the `OpenMP` *for loop* pragma to execute kernels in batched mode. The former raises concerns on software sustainability while the latter may not extract performance of the underlying hardware architecture.

The authors in [26] have introduced the algorithm of a single TLR-MVM to enhance real-time performance when identifying the atmospheric turbulence for ground-based telescopes. Based on batched MVM, their TLR-MVM implementations rely on `OpenMP` due to standardization constraints required in the computational astronomy community for code sustainability and portability purposes. Performance results have been reported on several cutting-edge architectures.

In this paper, we extend this previous work [26] to process multiple TLR-MVM (i.e., a batch of batched MVM) required by the seismic redatuming method and deploy the application to distributed-memory systems equipped with x86 nodes and vector computing engines. Given the few but strong cores (i.e., eight cores) on NEC SX-Aurora TSUBASA cards compared to x86 architectures, NEC hardware solution makes up the low core count with vectorizations and high bandwidth memory (HBM2) to achieve high performance. This is quite different than GPU architectures that promote massive parallelism via the single instruction, multiple data (SIMD) paradigm. Therefore, porting on NEC vector engines represents a similar effort than deploying on x86, i.e., with high user productivity, with the favorable exception that HBM2 may provide a significant performance boost to memory-bound kernels compared to x86's DDR4 memory technology. And to further maximize performance on NEC vector engines, it is also paramount to fully utilize the vector units. All in all, the hardware design of NEC SX-Aurora TSUBASA cards facilitates the deployment of these advanced numerical kernels, which intrinsically drives the performance of low-rank matrix approximations.

To our knowledge, this is the first time TLR-MVM is successfully applied to 3D datasets using NEC vector computing hardware solutions in the context of seismic redatuming.

2. Seismic Redatuming

Seismic redatuming is the process of numerically re-positioning seismic data physically recorded at the surface of the Earth to any location of interest in the subsurface. Whilst historically able to target only so-called primary arrivals in the recorded data, recent theoretical advances have led to the creation of so-called *Marchenko redatuming*, which is capable of handling full-wavefield seismic data including any order and type of internal scattering. This entails an inverse problem to be solved that can be expressed concisely as the following system of equations [27]:

$$\begin{bmatrix} \Theta \mathbf{R} \mathbf{f}_{\mathbf{d}}^{+} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\Theta \mathbf{R} \\ -\Theta \mathbf{R}^{*} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}^{-} \\ \mathbf{f}_{\mathbf{m}}^{+} \end{bmatrix}, \tag{1}$$

where $\mathbf{R}$ and $\mathbf{R}^{*}$ are so-called convolution and correlation integral operators, Θ is a time-space window, $\mathbf{I}$ is the identity operator, $\mathbf{f}^{-}$ and $\mathbf{f}_{\mathbf{m}}^{+}$ are the up-going and the coda of the down-going focusing functions to invert for, and $\mathbf{f}_{\mathbf{d}}^{+}$ on the left-hand side of Eq. 1 is the direct component of the down-going focusing function that can be obtained by numerical modelling in a reference velocity medium. Finally, the overall down-going focusing function can be created as $\mathbf{f}^{+} = \mathbf{f}_{\mathbf{d}}^{+} + \mathbf{f}_{\mathbf{m}}^{+}$. For simplicity, Eq. 1 can be written compactly as $\mathbf{d} = \mathbf{M}\mathbf{f}$, where $\mathbf{d}$, $\mathbf{f}$ and $\mathbf{M}$ are the overall data, model, and Marchenko operator of the problem we wish to solve.

Once the focusing functions are retrieved, the up- and down-going separated Green's functions $\mathbf{g}^-$ and $\mathbf{g}^+$ can be computed to be evaluating the following equations:

$$\begin{bmatrix} -\mathbf{g}^- \\ \mathbf{g}^{+*} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ -\mathbf{R}^* & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}^- \\ \mathbf{f}^+ \end{bmatrix}. \tag{2}$$

Note that due to the extremely large size of these matrices, whilst the problem is written in a compact matrix-vector formulation, its numerical solution is performed using matrix-free operators and iterative solvers such as conjugate gradient least-squares (CGLS) or LSQR [29]. Focusing now our attention on the multi-dimensional convolution (MDC) integral operator, which represents the most expensive computations in the overall chain of operations, its inner working can be written more explicitly as follows:

$$\mathbf{y} = \mathbf{R}\mathbf{x}: \quad y(t, \mathbf{x}_B, \mathbf{x}_A) = \mathcal{F}^{-1}_{\omega_{max}} \left(\int_{\delta\mathbb{D}} R(\omega, \mathbf{x}_B, \mathbf{x}_R) \mathcal{F}_{\omega_{max}} \left(x(t, \mathbf{x}_R, \mathbf{x}_A) \right) d\mathbf{x}_R \right). \tag{3}$$

Similarly, the adjoint of such an operator can be written as:

$$\mathbf{x} = \mathbf{R}^H\mathbf{y}: \quad x(t, \mathbf{x}_R, \mathbf{x}_A) = \mathcal{F}^{-1}_{\omega_{max}} \left(\int_{\delta\mathbb{D}} R^*(\omega, \mathbf{x}_B, \mathbf{x}_R) \mathcal{F}_{\omega_{max}} \left(y(t, \mathbf{x}_B, \mathbf{x}_A) \right) d\mathbf{x}_B \right), \tag{4}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the forward and inverse Fourier transforms, ω is the angular frequency, $\mathbf{x}_A$, $\mathbf{x}_B$ and $\mathbf{x}_R$ represent spatial locations with the latter two spanning the integration domain $\delta\mathbb{D}$. ω_{max} is used to indicate that the output of the forward Fourier transform is truncated to contain only frequencies where the signal spectrum resides. Finally, $R(\omega, \mathbf{x}_B, \mathbf{x}_R)$ represents the kernel of the integral operator in the frequency-space domain and can be created upfront by applying the Fourier transform along the time axis of the physically recorded seismic data $R(t, \mathbf{x}_B, \mathbf{x}_R)$. Moreover, once the spatial integral is discretized, the kernel simply becomes a stack of matrices (one for each frequency ω within the specified frequency spectrum of the seismic data) and the integral can be interpreted as a batched matrix-vector multiplication (MVM) operations. This is true for both the forward and adjoint computations, with the main difference that the latter requires such matrices to be transposed and complex conjugated. Finally, in order to validate our statement that the operators $\mathbf{R}$ and $\mathbf{R}^H$ represent the most expensive computations in the solution of the inverse problem in Eq. 1, a single iteration of CGLS is evaluated and the overall computational time is divided into atomic contributions (Fig. 1). A single-node implementation of the Marchenko redatuming equations is used in this example as provided by the PyLops framework for large-scale inverse problem [28]. More specifically, since an iteration of CGLS requires the application of both a forward ($\mathbf{M}$) and adjoint ($\mathbf{M}^H$) passes, we observe that almost ninety percent of the time is spent on evaluating the $\mathbf{R}$ (and $\mathbf{R}^*$) and their adjoints, while the remaining time percent of the time is roughly split between other computations involved in the $\mathbf{M}$ operator and vector-vector operations in the CGLS step. Finally, we observe here a slight time difference in the computation of the forward and adjoint passes; this results from the transposition of the matrix stack in the adjoint step. Moreover, complex conjugation must be performed on each element of the kernel. In practice, as explained in more detail in [29], complex conjugation is applied to the input and output vectors, which are much smaller than the kernel, so the kernel itself is not transposed. Nevertheless, since the matrices in the stack are stored in a row-major order in main memory, the timing of the forward step is more favourable.

Alongside with advances in processing algorithms, the size and scale of seismic surveys have increased since the late 20th century. Nowadays, large-scale high-resolution 3D surveys

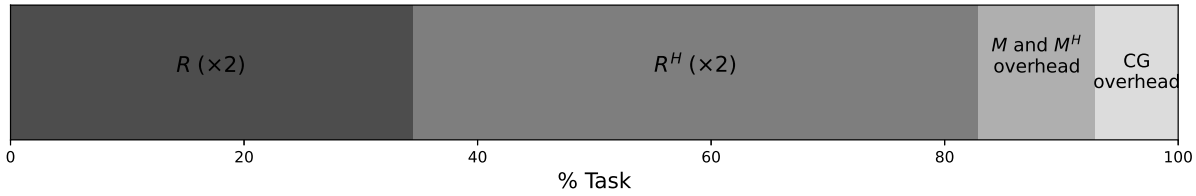


Figure 1. Task profiling for a single iteration of CGLS during the solution of the inverse problem in Eq. 1

are routinely acquired where the recorded data can easily be on the order of several Terabytes. Recently, the implementation of the 3D Marchenko equations has been discussed in [29] and [11]. In both cases, special attention has been placed on the implementation of the integral operator and the handling of kernels that cannot directly fit in the main memory of single compute node. In the former approach, the embarrassingly parallel nature of the batched MVM is leveraged by reading different frequency batches in the main memory of multiple compute nodes only once prior, to solving the inverse problem in Eq. 1. The latter approach, on the other hand, utilizes the ZFP-based compression algorithm of [25] to reduce the size of the reflection response to be stored on disk and the read-in time to memory. Authors report a compression factor of four for this lossless compression when applied to their frequency-space reflection seismic data. In fact, even when the data is compressed, on-the-fly decompression is still required to be able to perform the computations in Eqs. 3 and 4. In both implementations, however, no attempt is made to expedite the dense MVM required in both the forward and adjoint processes.

Whilst the redatuming Eq. 1 is relatively new to the field of geophysics, integral operators of the kind of $\mathbf{R}$ (Eq. 3) and $\mathbf{R}^H$ (Eq. 4) are common to a number of other wave-equation-based seismic processing methods, such as surface-related multiple elimination (SRME – [32]), estimation of primaries by sparse inversion (EPSI – [19]), and up/down deconvolution [6] just to name a few. This work may therefore have a broader impact beyond the Marchenko redatuming technique.

3. Background on TLR-MVM

We briefly recall here the algorithmic design of the tile low-rank matrix-vector multiplication (TLR-MVM) kernel [26]. While the traditional dense MVM stores the matrix elements in column-major data layout format, TLR-MVM first splits the dense matrix into tiles in which elements are now contiguous in memory. This tiling technique is important in terms of memory access as it shortens the strided memory access to better fit in the high levels of the memory subsystem. Once the tile matrix data structure is constructed, TLR-MVM compresses each dense tile in an embarrassingly parallel fashion using an algebraic method of choice (e.g., rank-revealing QR, randomized SVD, etc.). The numerical lossy compression depends on the accuracy threshold required by the application to sustain its numerical robustness.

Figure 2 represents the TLR-MVM operation $A \times x = y$, with $A \in \mathbf{R}^{m \times n}, x \in \mathbf{R}^n, y \in \mathbf{R}^m$. The matrix A is split into 6-by-7 tiles with a tunable tile size parameter nb. After compression using the accuracy threshold ϵ, each tile is decomposed into U and V bases containing the k most significant singular values $\Sigma_{i,j}$ and their associated singular vectors, as defined by the following formula based on the Frobenius norm: $||A_{i,j} - U^{\epsilon}_{i,j}\Sigma^{\epsilon}_{i,j}V^{T\epsilon}_{i,j}||_F \leq \epsilon||A||_F$. The selected singular values $\Sigma_{i,j}$ may be absorbed by one of the bases after applying corresponding scaling

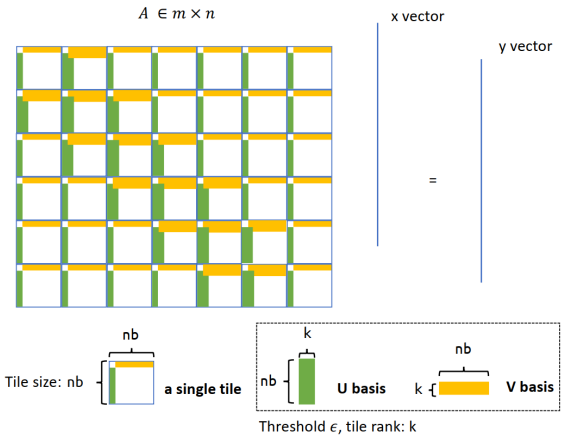


Figure 2. TLR-MVM data structure

operations, which facilitates the MVM implementation. Once compressed, the tiles may have different ranks k, which may create load imbalance situations.

Moreover, Level-2 BLAS MVM kernels are inherently memory-bound and their performance solely depends on the memory bandwidth. Therefore, it is critical to optimized memory access to avoid additional data motion between main memory and the last-level cache. While tiling technique helps for the traditional dense MVM, TLR-MVM ends up dealing with several new compressed U and V data structures that may not be stored contiguously in memory. As introduced in [26], we stack the U bases together and the V bases together and design a new MVM algorithm that leverages the TLR data structure.

The actual TLR-MVM operation can then proceed with the following three successive computational phases: (1) we multiply the stacked V bases to the vector x, (2) we project the result from phase 1 to the V bases, and (3) we multiply the stacked U bases with the result from the reshuffling phase 2 and compute the final result vector y.

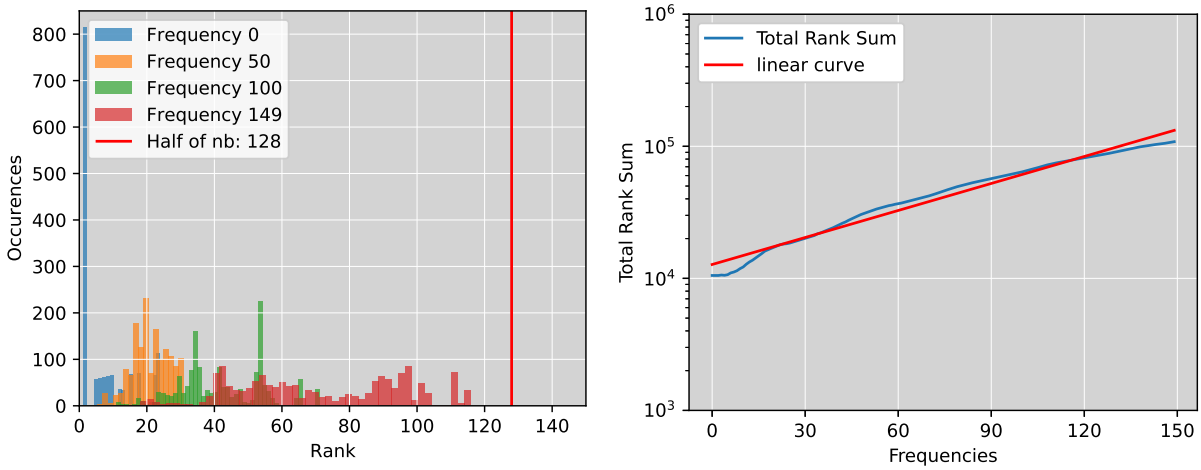
In this paper, we extend the real precision arithmetic used for computational astronomy [26] to complex precision arithmetic in order to support seismic imaging applications. Furthermore, we implement a driver to launch several TLR-MVM kernels, i.e., one for each frequency ω, coming from the discretization of the spatial integral $R(\omega, \mathbf{x}_B, \mathbf{x}_R)$ (see Section 2), as explained in the next section.

4. Launching Multiple TLR-MVM Kernels for Seismic Redatuming

In this section, we describe the implementation of a driver that launches multiple TLR-MVM kernels to support the workload of seismic redatuming. We first identify the challenges introduced by TLR-MVM on 3D seismic datasets and propose optimization techniques to address them.

4.1. Challenges with 3D Seismic Datasets

The 3D seismic dataset used in this study contain stacked matrices for 150 frequencies. This is representative of the workload of seismic redatuming, although the number of frequencies may be further increased to include higher frequencies to produce seismic wavefields and images of higher resolution. The size of each matrix is $m = n = 9801$. The matrices considered herein



(a) Rank distribution

(b) Rank summation for each frequency matrix

Figure 3. Rank analysis of the 3D seismic dataset using $nb = 256$ and $\epsilon = 0.001$

are square, but rectangular matrices are also supported. The relationship between the frequency index F_i and the value of the frequency is given by $f_{F_i} = \frac{i}{d_t n_t}$, where $d_t = 0.0025$ and $n_t = 1201$.

Figure 3 shows the rank analysis of the 3D seismic datasets. In particular, Fig. 3a highlights the rank distribution of F_i matrices for i = 0, 50, 100 and 149. We set $nb = 256$ and $\epsilon = 0.001$. As expected, the figure captures how the rank distribution shifts to the right with higher ranks for matrices corresponding to higher frequency components of the data. The red vertical line in Fig. 3b shows the rank limit $nb/2 = 128$. If the rank distribution goes beyond this red line, the accumulated sizes of the bases for a single tile will be higher than the size of the original dense tile. On the contrary, if the rank distribution stays on the left of the red line, as pictured in Fig. 3a, TLR-MVM remains competitive compared to dense MVM. Figure 3b reports the summation for all ranks for a given frequency matrix. The total rank summation is a good metric to evaluate the algorithmic complexity. We observe an increase in rank for higher frequencies, which corroborates the analysis of the rank distribution in Fig. 3a. For this particular 3D seismic dataset, we can observe from Fig. 3b that the log-scale of the number of floating-point operations (FLOPS) of TLR-MVM has a near-linear relationship with the frequency matrix index. This relationship provides insights on how to orchestrate the TLR-MVM scheduling for all frequencies, given the workload heterogeneity. It is now clear that one of the main challenges is the load imbalance introduced by TLR-MVM within and across all frequency matrices, compared to the homogeneous dense MVM. We implement two optimizations techniques and present their corresponding pseudo-codes in Fig. 4. The codes are written in C and rely on `MPI+OpenMP` programming models.

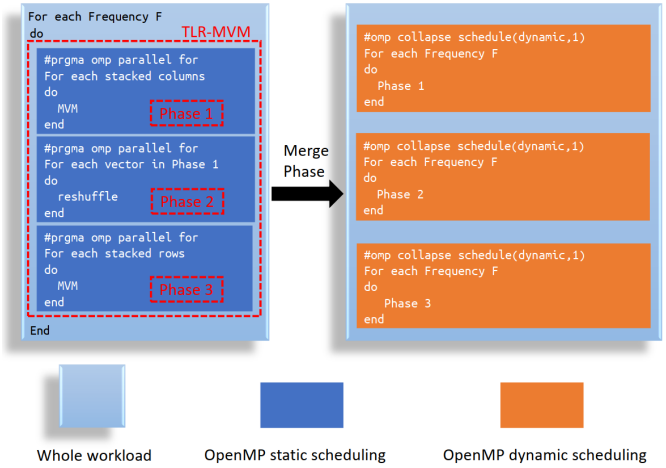
We address in subsequent sections how these techniques mitigate the load imbalance overhead.

4.2. Merge-Phase Strategy for Intra-Node Load Balancing

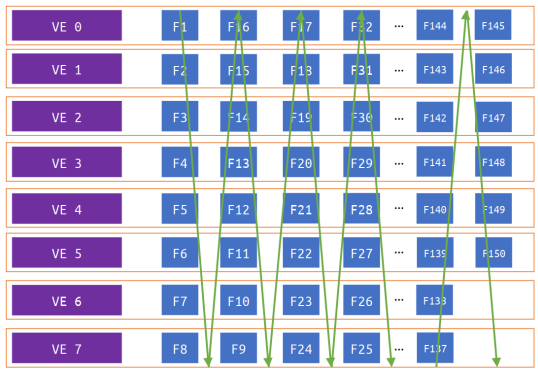
Figure 4a pictures the *Merge-Phase* strategy (in orange color) proposed for our TLR-MVM reference implementation (in blue color).

The *Merge-Phase* strategy is designed to achieve intra-node load balancing by evenly distributing the computation on each thread (or processing units). Processing a collection of fre-

quencies F is not performed one by one anymore. Instead, the strategy fuses each individual phase across all frequencies. This increases the workload per `OpenMP` loops within each phase, engenders a larger amount of computational tasks, and reduces the scheduling overheads. The default static scheduling mode of operation may increase data locality, especially when dealing with small datasets. For large datasets, static scheduling may lose its performance advantage and may create idle time while work is available. We further enable the `OpenMP` dynamic scheduling so that the runtime has opportunities to prevent idle time situations by scheduling tasks as soon as they enter ready state. While this strategy alleviates the intra-node load imbalance bottleneck, the inter-node load imbalance remains an issue, as observed in Fig. 3b.



(a) The Merge-Phase strategy



(b) The ZigZag mapping strategy using eight vector engines (VE) and 150 frequency matrices F_i with i = 0 , ... , 149

Figure 4. Pseudo-codes of the *Merge-Phase* and the *ZigZag* mapping strategy

4.3. ZigZag Mapping Strategy for Inter-Node Load Balancing

To leverage performance on distributed-memory systems, we evenly allocate frequency matrices across computational nodes. Dynamic load balancing on distributed-memory systems is a challenging approach that may require data movement to compensate the for the idle time. However, we have identified some relationship between the frequency index and the corresponding FLOPS, as explained in Section 4.1. The *ZigZag* mapping strategy is used to achieve inter-node load balancing. We design the *ZigZag* mapping strategy to statically map frequencies to processing units and achieve inter-node load balancing. Figure 4b highlights the *ZigZag* pattern for frequency mapping using 150 frequencies on eight Vector Engines (VE).

In the first sweep of VEs, we map the frequencies in increasing index order. We continue mapping the next set of frequencies in decreasing index order for the VEs. We repeat the above *ZigZag* pattern until all frequencies are mapped to the VEs. This strategy has several advantages. Not only does it exploit the linear relationship between frequency index and FLOPS, it also balances the memory footprint per VE. For instance, given that the on-chip memory capacity is limited to 48 GB on NEC VEs, the *ZigZag* mapping strategy allows to scale memory-intensive applications. Furthermore, this mapping is performed offline and does not incur runtime overheads. Although it is important to mention that this linear relationship may not be characteristic of all seismic datasets. Network interconnect congestions may even further exacerbate the load

imbalance even if the load is properly distributed. Therefore, we believe dynamic load balancing is an interesting research direction and we leave it as future work.

4.4. Algorithmic Complexity and Code Balance

As discussed in [26], the number of floating-point operations (FLOPS) of the dense MVM is $2mn$ and the memory bandwidth can be calculated as $\frac{B(mn+n+m)}{t}$, with $B(W)$ the number of bytes of W elements (stored in single complex precision) and t the execution time. The FLOPS and memory bandwidth of TLR-MVM are $4K \times nb$ and $\frac{B(2K \times nb+4K+m+n)}{t}$, respectively, with K the sum of the ranks across all tiles of any single frequency matrix (see Fig. 2) and nb the tile size.

In seismic application, suppose there are F frequency matrices, the overall FLOPS and memory bandwidth of dense MVM is $F \times 2mn$ and $F \times \frac{B(mn+n+m)}{t}$, respectively. The overall FLOPS and memory bandwidth of TLR-MVM for all frequency matrices F is $4K_F \times nb$ and $\frac{B(2K_F \times nb+4K_F+m+n)}{t}$, where K_F is the sum of all ranks across all frequencies. According to the FLOPS calculation of TLR-MVM, the rank sum K of the frequency matrix plays an important role. If the rank sum K is not large enough, the algorithm may not saturate the memory bandwidth due to a suboptimal hardware occupancy.

5. Numerical Accuracy Assessment

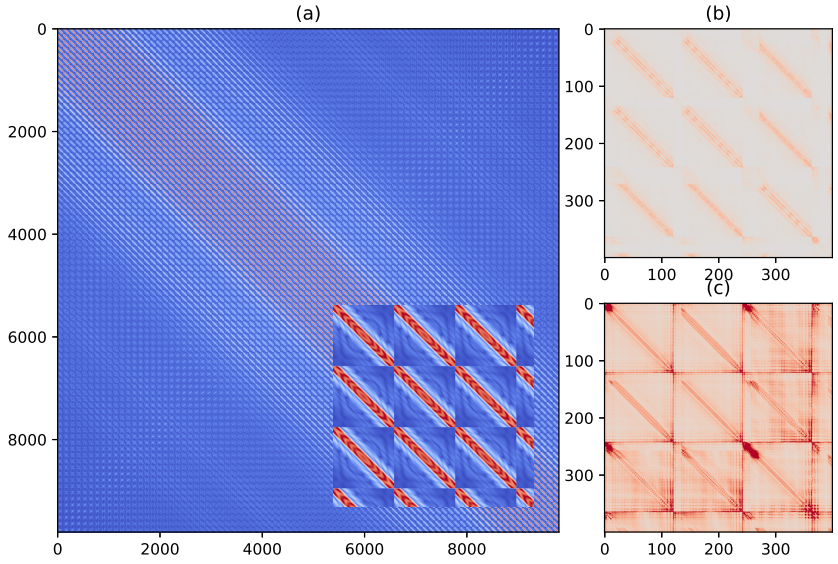


Figure 5. (a) Original Matrix of reflection response frequency slice at 33 Hz ($R(\omega = 33Hz, \mathbf{x}_B, \mathbf{x}_R)$, Insert is zoomed-in version on the first 400 rows and columns (b) and (c) show the same zoomed-in version of TLR compression error in (a), nb is tile size, ϵ is accuracy

To begin with, we consider the same synthetic geological model used in [29] (their Fig. 3) and compare the reconstruction of the total Green's function $\mathbf{g} = \mathbf{g}^+ + \mathbf{g}^-$ obtained using the original R kernel and the TLR compressed R kernel with different combinations of tile size and accuracy. The observation that seismic data in the frequency domain can be compressed in a low-rank form is not surprising when considering the form of matrices representing seismic data in the frequency domain as shown in Fig. 5a. Such a matrix represents the seismic data at a single frequency ($f = 33\ Hz$) with sources placed along the rows and receivers along with the

columns. In other words, the element i, j of this complex-valued matrix contains the frequency at 33 Hz coefficient of the seismic recording at $i-th$ source and $j-th$ receiver. Looking at the insert in Fig. 5a, we can see how the entire matrix is composed of smaller diagonally dominant submatrices. This is due to the fact that we are dealing with 3D seismic data and each submatrix represents the responses of a single source line to a single receiver line. Moving from one source line to the next leads to the block pattern of this matrix in the row space, whilst moving from one receiver line to the next leads to the block pattern in the column space. Another important observation that was made by [22] is that within the available seismic bandwidth, kernel matrices at higher frequencies are of higher rank, i.e., require more basis functions to be approximated at the selected accuracy. Whilst not discussed by the authors in [22], this imbalance in the rank of the different matrices inevitably leads to a load imbalance in the computation of the batch matrix-vector multiplications (MVM), as explained in Section 4.1. The effect of solving the inverse problem in Eq. 1 using different TLR compressed R kernels is presented in Fig. 6. More specifically, Fig. 6a shows the reconstructed Green's function along eight different receiver lines, as shown in the insert using the original uncompressed reflection response. Such an estimate has been shown in [29] to be very accurate and is used here as our benchmark. Figures 6b-d show the reconstructed Green's function using TLR-MVM with different compression parameters. We choose $nb = 256, \epsilon = 0.001, 0.005, 0.01$. Figure 6e shows element-wise absolute value difference between Fig. 6a and Fig. 6b. Figure 6f is $10\times$ of the value in Fig. 6e. Figures 6g-h are also $10\times$ the element-wise absolute value difference with Fig. 6a using corresponding compression parameters.

We use Signal-to-Noise Ratio (SNR) to quantify the error shown in Fig. 6. The formula is $\text{SNR} = -20 * \log_{10} \frac{\|R_{org} - R_{approx}\|_2}{\|R_{approx}\|_2}$: the larger the SNR, the better the approximation.

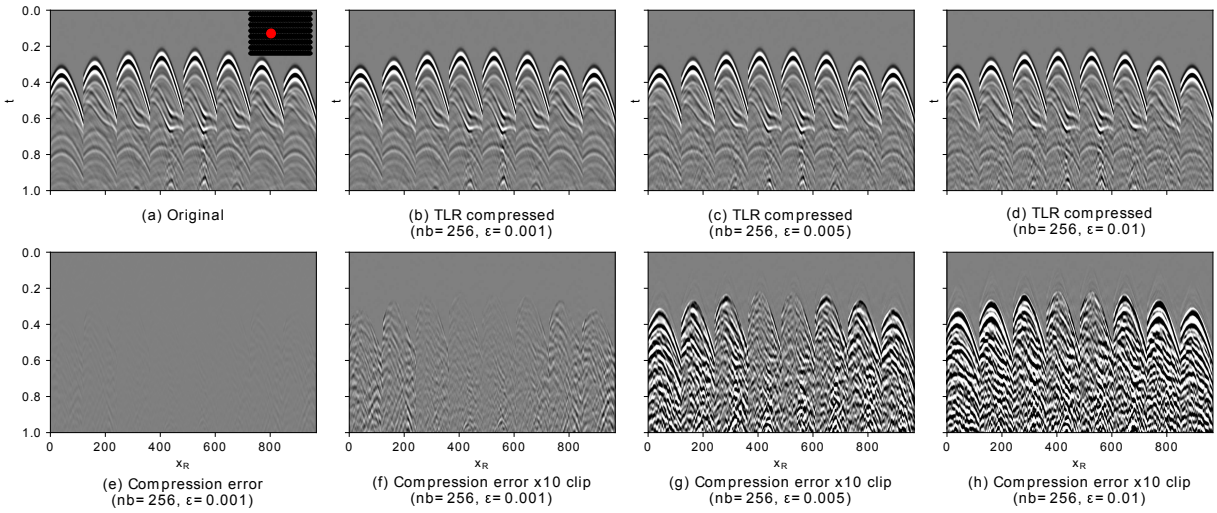
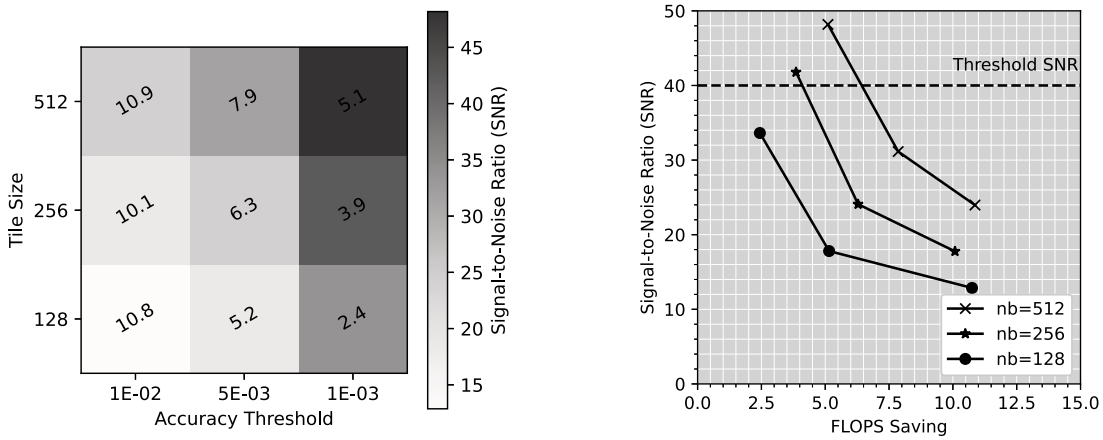


Figure 6. Green's function estimates (t^2 gain applied to all panels), with nb tile size and ϵ the accuracy threshold

Next, we investigate the impact of compression on the SNR for various tile sizes and accuracy thresholds, as shown in Fig. 7a. We compute the ratio between the FLOPS executed by dense MVM versus TLR-MVM. This ratio of FLOPS savings corresponds to how much fewer FLOPS are performed by TLR-MVM compared to dense MVM. For instance, TLR-MVM achieves an SNR around 30 while performing 2.4 fewer FLOPS than dense MVM when using $nb = 128$ and $\epsilon = 0.001$. We use colormap to show the corresponding SNR value. There is a general trend that one needs a more restrictive accuracy threshold in cases with smaller tile sizes. Figure 7b



(a) (b)

Figure 7. (a) Heat map plot of Signal-to-Noise Ratio (SNR), against tile size nb, accuracy threshold ϵ, and FLOPS saving (b) Line plot of SNR versus FLOPS savings for different tile sizes

shows the relationship of FLOPS savings with SNR. We choose a threshold SNR value of 40 that satisfies the quality requirement of the application. We observe two sets of compression parameters ($nb = 256, \epsilon = 0.001$ and $nb = 512, \epsilon = 0.001$), which satisfy this requirement. Indeed, under these two compression parameters, TLR-MVM algorithm does not affect the final image quality.

Once we complete the assessment on the numerical accuracy with the identification of this couple of sets of parameters, we can now study their impact on performance with the hardware systems studied in this paper.

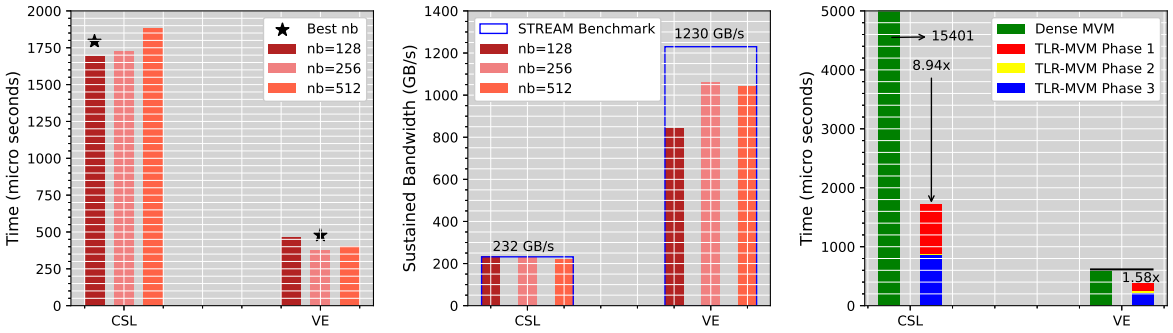
6. Experimental Results

This section reports the performance results of our TLR-MVM implementation, compares it against dense MVM, and highlights the impact of the optimization techniques introduced in Sections 4.2 and 4.3.

6.1. Environment and Compilation Settings

The experiments are carried out on two x86 systems. The first system is a 16-node NEC B300-8 cluster, where each node is equipped with 8 NEC SX-Aurora TSUBASA-20B Vector Engines (VE). The memory capacity of each VE is 48GB. The second system is a single x86 node with a dual-socket 20-core Intel Cascade Lake. We refer to this shared-memory node to CSL in the subsequent sections. The memory capacity of the Intel server is 350 GB. The OS of both systems is Linux RedHat 7.7. We use OpenMPI as the MPI implementation. For the NEC server, we use NEC Compilers tools to compile the code and rely on the NEC Numeric Library Collection for the vendor optimized BLAS implementation. For the Intel server, we use Intel Parallel Studio 2019 to compile the code and link it against Intel Math Kernel Library for the vendor optimized BLAS implementation. We use -fopenmp -O3 for both compilations. To analyze the performance results, we rely on NEC `Ftrace` profile analysis tool. All experiments are run in single complex precision arithmetics.

6.2. Performance Results on Synthetic Datasets



(a) Time to solution (b) Memory bandwidth (c) Time breakdown

Figure 8. Performance analysis of TLR-MVM on synthetic datasets

We first study TLR-MVM on synthetic datasets to demonstrate the robustness and software capabilities of our TLR-MVM implementation. We randomly generate U and V bases in single complex precision. Without loss of generality, we employ square matrices with $m = n = 10000$ because it is closer to the real seismic datasets. The rank k is set to $nb/4$ and is constant across all tiles. This specific rank translates into a theoretical FLOPS saving factor of 2, i.e., TLR-MVM performs twice fewer FLOPS than dense MVM. Figures 8a and 8b show the time to solution and memory bandwidth using the synthetic datasets, respectively, on the single Intel CSL node and one NEC VE. We test three tile sizes, i.e., $nb = 128, 256, 512$. We also show the memory bandwidth obtained by `STREAM` Benchmark. This is the upper limit of sustained memory bandwidth from DDR4 memory (i.e., Intel CSL) and HBM2 (i.e., NEC VE). For Intel CSL, we run one MPI process per socket with 20 `OpenMP` threads. For the single NEC VE, we run one MPI process with 8 `OpenMP` threads. We see that a single NEC VE is much faster than Intel CSL thanks to HBM2 technology with up to 85 % of bandwidth saturation. On NEC VE, the configuration with the best time / bandwidth (annotated with a star) runs in less than 375 micro seconds and exceeds 1 TB/s.

Figure 8c shows the time breakdown of the three phases in TLR-MVM on Intel CSL and one NEC VE. As expected, the computational phases 1 and 3 are the most time-consuming with the batched MVM capturing most of the elapsed time. We also compare TLR-MVM with dense MVM, as implemented in the Level-2 BLAS `CGEMV` kernel in the vendor optimized libraries. TLR-MVM reaches 8.94 and 1.58 performance speedups compared to dense MVM on Intel CSL and one NEC VE, respectively. Compared to the theoretical FLOPS saving factor of 2, our TLR-MVM implementation gets higher performance speedup on Intel CSL thanks to a full saturation of the memory bandwidth. The dense MVM implementation from Intel MKL may not saturate the bandwidth enough and it seems to be poorly optimized. On the NEC VE card, our TLR-MVM implementation achieves less performance speedup when compared against the dense vectorized MVM from NEC NLC. As shown in Fig. 8b, there may still be some room for improvement for our TLR-MVM implementation in further saturating the main memory.

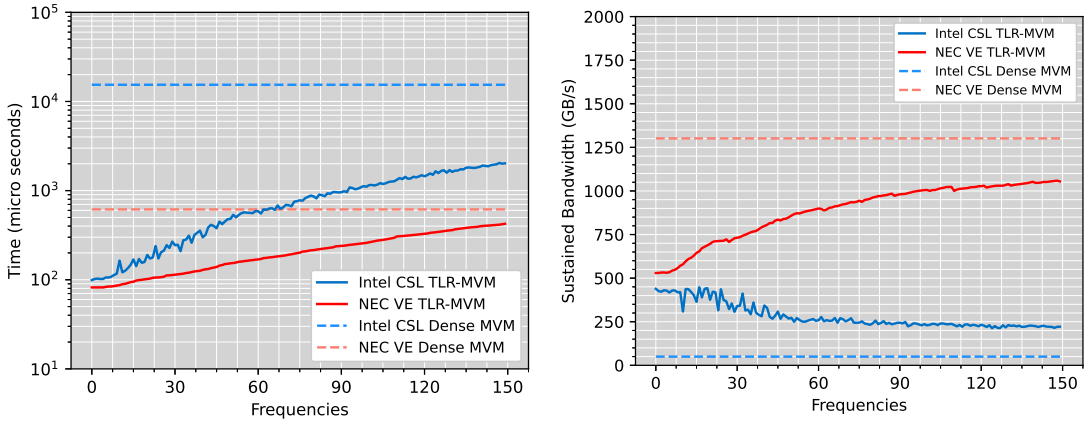
6.3. Performance Results on 3D Seismic Datasets

In this section, we run against 3D seismic datasets and conduct experiments using 150 frequencies. Each frequency matrix is of size 9801×9801. As identified in Section 5, we use $nb = 256$

and $\epsilon = 0.001$ to compress the dense matrix into a tile low-rank (TLR) matrix, while delivering application's accuracy *and* performance on the NEC VE-based system.

6.3.1. Performance analysis over all frequencies

Figure 9a compares the time to solution for each frequency matrix on Intel CSL and NEC VE systems when running the dense MVM and TLR-MVM. The time for dense MVM is constant across all frequencies on both systems. However, the time for TLR-MVM increases with the frequency index. This trend confirms the outcome of the rank statistics analysis made in Section 4.1, where the ranks (i.e., the workloads) grow with the index of the frequency matrix. For high frequencies, TLR-MVM on NEC VE outperforms its counterpart on Intel CSL by up to a factor 4, which is aligned with results on synthetic datasets from Section 6.2. Figure 9b shows the same performance analysis but with the sustained bandwidth obtained for each frequency matrix on the two systems. The TLR-MVM bandwidth increases with the frequency index and achieves more than 1 TB/s for high frequencies. On Intel CSL, the dense MVM implemented in `CGEMV` kernel from MKL shows limited sustained bandwidth while TLR-MVM on the same system is able to saturate the bandwidth, as already demonstrated for synthetic datasets in Section 6.2. However, there is a clear load imbalance issue when looking at all frequencies and their respective makespan.



(a) Load imbalance in time

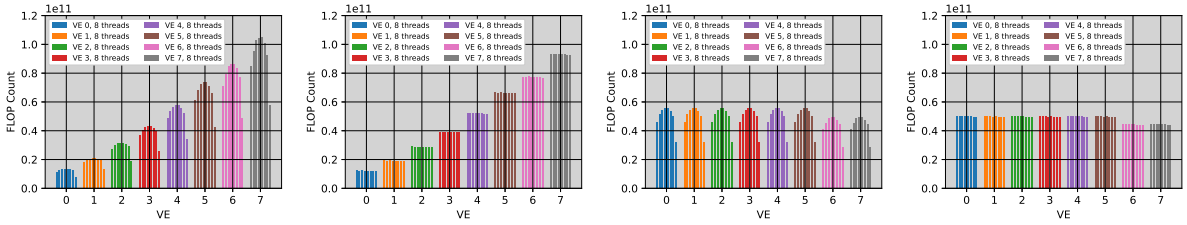
(b) Load imbalance in bandwidth

Figure 9. Performance analysis (time and bandwidth) for all frequency matrices (stored in dense and TLR) on Intel CSL and NEC VE systems, using $nb = 256$ and $\epsilon = 0.001$ for compression

6.3.2. Performance impact of optimization techniques

In this section, we assess the performance impact of the two previously introduced optimization techniques from Sections 4.2 and 4.3. We investigate four scenarios to solve the load imbalance issue: the reference TLR-MVM, the Merge-Phase TLR-MVM, the reference TLR-MVM with ZigZag mapping strategy and the Merge-Phase TLR-MVM with ZigZag mapping strategy. We conduct experiments on the 3D seismic dataset with 150 frequencies on 8 NEC VEs. We use NEC `Ftrace` analysis to get the FLOPS count of each `OpenMP` thread to illustrate how the various strategies can mitigate the load imbalance overhead. The default mode of NEC `Ftrace` is Vector Operation profiling. We need to set the environment variable VE_PERF_MODE to VECTOR-MEM to get the FLOPS count. Figure 10 reports the FLOPS count of each thread

for the aforementioned scenarios. Figure 10b shows how the Merge-Phase strategy improves the load balancing among threads within each VE. Figure 10c highlights how the ZigZag mapping strategy further distributes evenly the workload between NEC VE cards. Figure 10d pictures the performance impact when both strategies are activated. It permits to achieve inter-node and intra-node load balancing. It is also noteworthy to mention that the workload among threads



(a) Reference (b) Merge-Phase (c) ZigZag mapping (d) both (b) and (c)

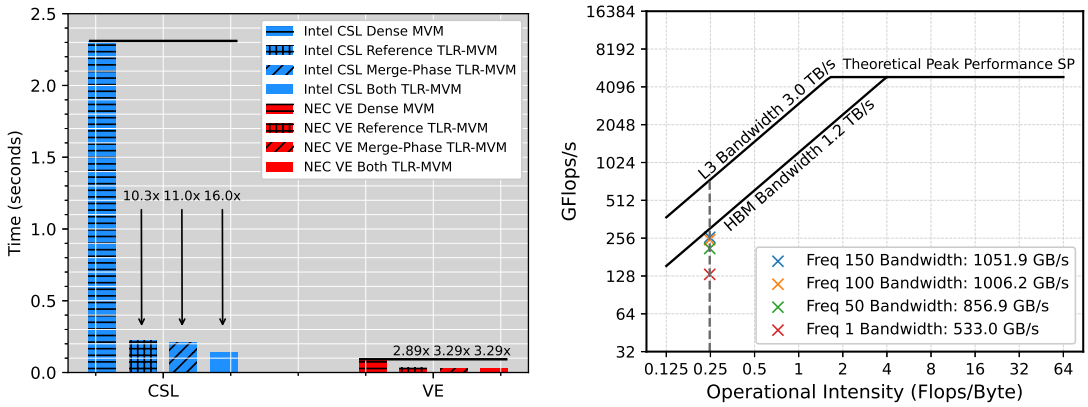
Figure 10. FLOPS count of different load balancing optimization technique for all threads when running against 150 frequencies using 8 VEs, (d) uses both Merge-Phase and ZigZag Mapping optimization

from VE 0 to VE 5 is higher than VE 6 and 7. This is because the number of frequencies (i.e., 150) is not divisible by the number of VEs (i.e., 8). Therefore, the first 6 VEs end up having one more frequency matrix to process.

Figure 11a compares time to solution of our TLR-MVM implementation with various optimization techniques against vendor optimized dense MVM on the overall application using one single Intel CSL node and a single NEC VE. TLR-MVM achieves up to 16X performance speedup against dense MVM on Intel CSL. This speedup factor is due to a suboptimal implementation of MKL multithreaded `CGEMV` kernel. Moreover, TLR-MVM scores up ot 3.3X performance speedup compared to dense MVM on a single NEC VE. This result is on par with the theoretical FLOPS saving factor of 3.9 reported in Fig. 7a. The optimization techniques do not impact significantly the TLR-MVM performance variants on the single NEC VE since the number of `OpenMP` threads is limited to 8. However, they do impact the TLR-MVM performance on the Intel CSL system, due to the large thread count, i.e., a total of 40 threads.

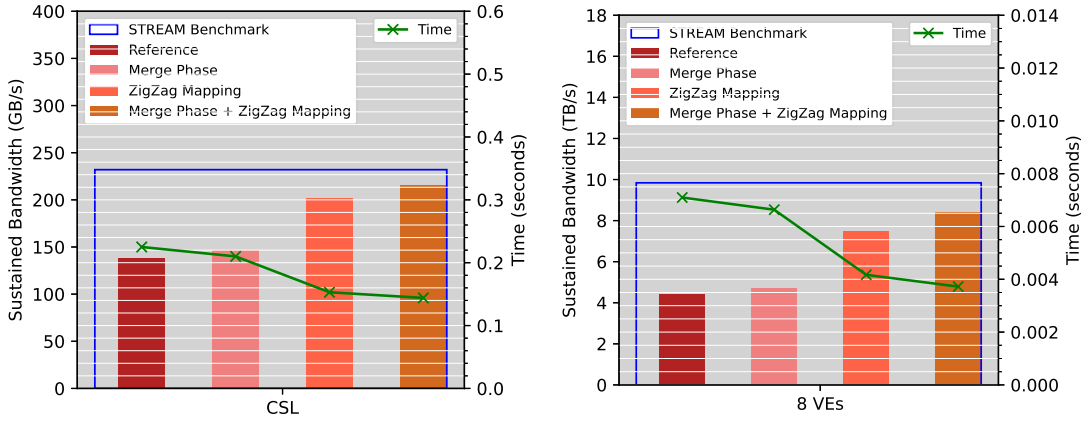
Figure 11b shows the roofline performance model using a single NEC VE, while combining Merge-Phase + ZigZag mapping strategies. We select frequency indices 1, 50, 100, and 150 and show how TLR-MVM performance increases with the frequency index and gets near the HBM2 sustained bandwidth. Although the gain in time is important, the fine-grained computation of TLR-MVM may prevent matrices with low frequencies from getting closer to the sustained bandwidth on the NEC VE system due to low vector units utilization.

Figures 12a and 12b show the bandwidth (left y-axis) and time to solution (right y-axis) for the overall simulation with 150 frequency matrices. The figures report performance for each of the four strategies on Intel CSL and 8 NEC VEs. By combining the Merge-Phase + Zigzag strategies, we achieve the best time to solution and over 85 % memory bandwidth of the STREAM Benchmark for both systems, thanks to a better hardware occupancy. Our TLR-MVM achieves around 9 TB/s aggregated bandwidth on 8 NEC VEs: this bandwidth score is approximately equivalent to 36 Intel CSL nodes. This result highlights the performance advantage of HBM2 over DDR4 memory technology.



(a) TLR-MVM Vs Dense-MVM (b) Roofline performance model

Figure 11. Performance comparison and assessment of TLR-MVM on Intel CSL node and single NEC VE



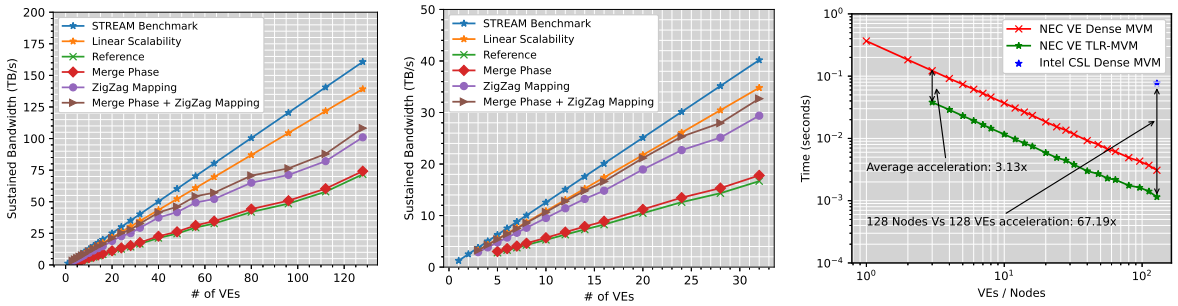
(a) Bandwidth/time on Intel CSL (b) Bandwidth/time on 8 NEC VEs

Figure 12. Performance impact of optimization techniques on TLR-MVM

6.3.3. Performance scalability

We scale up the number of VEs as well as the problem size to study the performance scalability of our TLR-MVM implementation with both Merge-Phase and ZigZag mapping strategies activated. In order to prove the scalability of our algorithm even beyond the 150 frequency matrices previously used, 3 additional fictitious matrices are created in between each of the available matrices by means of linear interpolation. This leads to an augmented dataset composed of 596 frequency matrices, which is on par with real seismic applications that deal with broadband data (i.e., data spanning a broad range of frequencies). Figure 13a shows scalability results on 596 frequency matrices up to 128 NEC VEs. Our TLR-MVM implementation reaches 67 % of the sustained bandwidth and 77.7 % of linear scalability when using 128 NEC VEs. Figure 13b is a close-up view for up to 32 VEs. Note that the reference and Merge-Phase strategies can only run starting from 5 VEs because there is not enough memory on the VEs to host all frequency matrices. The ZigZag mapping strategy alleviates this bottleneck and permits to balance the memory allocation as well. Figure 13c shows time to solution comparison of vendor optimized dense MVM `CGEMV` kernel against TLR-MVM on up to 128 VEs. The average acceleration of TLR-MVM over dense MVM is 3.13× on NEC VEs. This number is again on par with the theoretical FLOPS saving factor of 3.9 reported in Fig. 7a, since the interpolation used

to extend the 3D seismic datasets is linear. Comparing against `CGEMV` from MKL on 128 dual-socket 20-core Intel Cascade Lake nodes with DDR4 memory, our TLR-MVM implementation achieves 67X performance speedup when using the same number of NEC VEs, i.e., 128 cards. This corresponds to an aggregated bandwidth around 110 TB/s.



(a) Performance scalability (b) Zoom-in (c) Performance comparisons

Figure 13. Performance scalability up to 128 NEC VEs

7. Discussions and Future Work

In this work, our attention has primarily focused on reducing the memory footprint of the kernel of the multi-dimensional convolution operator as well as improving its computational efficiency. Both goals have been largely achieved by means of the proposed TLR-MVM implementation. Given the nature of the inverse problem that we wish to solve (Eq. 1), several questions remain to be answered in our future endeavors. First, whilst we currently focus on improving the inner working of the $\mathbf{R}$ operator, the algorithm requires four slightly different computations to be implemented, namely $\mathbf{R}$, $\mathbf{R}^*$, $\mathbf{R}^*$, $(\mathbf{R}^*)^H = \mathbf{R}^T$. The latter three computations call for the application of the transpose and/or complex conjugate kernel to the input vector. Similar to its dense counterpart [29], we envision a TLR-MVM implementation where complex conjugation is applied to the input and output vectors instead of to the kernel itself. Moreover, given that the overall kernel is divided into tiles, the application of the transposed kernel is expected to benefit from the fact that elements of different rows of the U and V bases are closer in memory than their dense counterparts. Whilst this suggests a similar workload for the forward and adjoint passes, numerical validation is required. Moreover whilst our focus has so far not included the forward and inverse FFTs that comprise part of the operator in Eq. 3, future research will investigate the possibility to begin their computations as soon as some of the TLR-MVM have been executed without waiting for the computations over the entire frequency range to be finalized. Another opportunity lies in the fact that the inverse problem we wish to solve can be slightly modified to include more than one spatial coordinates $\mathbf{x}_B$ at the time [29]; in other words, our batched TLR-MVM can be replaced by a batched tile low-rank matrix-matrix multiplication (TLR-MMM) where each column of the input matrix represents the wavefield originated from a different virtual source.

Conclusion

In this paper, we investigate and deploy the Tile Low-Rank (TLR) Matrix-Vector Multiplication (MVM) performance to accelerate 3D seismic application workloads using vector computing hardware solutions based on NEC SX-Aurora TSUBASA architecture. We propose

and implement strategies to mitigate the load imbalance overheads that inherently emerge from such workloads. Our TLR-MVM implementation not only improves the overall performance but also permits to scale up in terms of memory footprint. Thanks to its fine-grained computations and memory-friendly data layout, our TLR-MVM implementation can leverage the HBM2 technology of NEC vector engines, which translates into a performance boost compared to Intel CSL architecture with DDR4 memory technology. We assess accuracy of our TLR matrix approximations and demonstrate the numerical robustness of our method by investigating the impact of compression on the subsurface image quality using signal-to-noise ratio as a qualitative metric. We then employ the roofline performance model to show how TLR-MVM is able to effectively extract performance from the underlying architecture. On distributed-memory environment, our TLR-MVM implementation reaches around 110 TB/s of aggregated bandwidth on 128 NEC vector engines, which converts to 67X perfromance speedup in time against vendor optimized dense MVM (i.e., MKL `CGEMV` kernel) with the same number of Intel CSL nodes. We believe these results are promising in the context of 3D seismic imaging. TLR-MVM may enable to increase the problem sizes further and eventually improve the quality of 3D land surveys.

Acknowledgements

For computer time, this research used the *Ibex* system hosted at the Supercomputing Laboratory at KAUST.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Abdelfattah, A., Haidar, A., Tomov, S., Dongarra, J.J.: Performance, design, and autotuning of batched GEMM for GPUs. In: Kunkel, J.M., Balaji, P., Dongarra, J.J. (eds.) High Performance Computing. ISC High Performance 2016. Lecture Notes in Computer Science, vol. 9697, pp. 21–38. Springer (2016). https://doi.org/10.1007/978-3-319-41321-1_2

2. Akbudak, K., Ltaief, H., Mikhalev, A., Keyes, D.: Tile Low Rank Cholesky Factorization for Climate/Weather Modeling Applications on Manycore Architectures. In: High Performance Computing. ISC 2017. Lecture Notes in Computer Science, vol. 10266, pp. 22–40. Springer (2017). https://doi.org/10.1007/978-3-319-58667-0_2

3. Akbudak, K., Ltaief, H., Mikhalev, A., *et al.*: Exploiting data sparsity for large-scale matrix computations. In: Aldinucci, M., Padovani, L., Torquati, M. (eds.) High Performance Computing. ISC High Performance 2016. Lecture Notes in Computer Science, vol. 11014, pp. 721–734. Springer (2018). https://doi.org/10.1007/978-3-319-96983-1_51

4. Al-Harthi, N., Alomairy, R., Akbudak, K., *et al.*: Solving Acoustic Boundary Integral Equations Using High Performance Tile Low-Rank LU Factorization. In: High Performance Computing. ISC High Performance 2020. Springer (2020). https://doi.org/10.1007/978-3-030-50743-5_11

5. Amestoy, P., Ashcraft, C., Boiteau, O., *et al.*: Improving Multifrontal Methods by Means of Block Low-Rank Representations. SIAM Journal on Scientific Computing 37(3), A1451–A1474 (2015). https://doi.org/10.1137/120903476

6. Amundsen, L.: Elimination of Free-surface Related Multiples Without Need of a Source Wavelet. Geophysics 66, 327–341 (2001). https://doi.org/10.1190/1.1444912

7. Berryhill, J.R.: Wave-equation Datuming Before Stack. Geophysics 49, 2064–2066 (1984). https://doi.org/10.1190/1.1441620

8. Börm, S.: Efficient Numerical Methods for Non-Local Operators: H2-matrix Compression, Algorithms and Analysis, vol. 14. European Mathematical Society (2010). https://doi.org/10.4171/091

9. Börm, S., Grasedyck, L., Hackbusch, W.: Introduction to Hierarchical Matrices with Applications. Engineering Analysis with Boundary Elements 27(5), 405–422 (2003). https://doi.org/10.1016/S0955-7997(02)00152-2

10. Boukaram, W.H., Turkiyyah, G., Ltaief, H., Keyes, D.E.: Batched QR and SVD Algorithms on GPUs with Applications in Hierarchical Matrix Compression. Parallel Computing 74(C), 19–33 (2018). https://doi.org/10.1016/j.parco.2017.09.001

11. Brackenhoff, J., Thorbecke, J., Koehne, V., *et al.*: Implementation of the 3D Marchenko method (2020). https://doi.org/10.1190/geo2017-0108.1

12. Broggini, F., Snieder, R., Wapenaar, K.: Focusing the Wavefield Inside an Unknown 1D Medium: Beyond Seismic Interferometry. Geophysics 77(5), A25–A28 (2012). https://doi.org/10.1190/geo2012-0060.1

13. Cao, Q., Pei, Y., Akbudak, K., *et al.*: Extreme-Scale Task-Based Cholesky Factorization Toward Climate and Weather Prediction Applications. In: Proceedings of the Platform for Advanced Scientific Computing Conference. pp. 2:1–2:11. ACM (2020). https://doi.org/10.1145/3394277.3401846

14. Charara, A., Keyes, D., Ltaief, H.: Tile Low-Rank GEMM Using Batched Operations on GPUs. In: Aldinucci, M., Padovani, L., Torquati, M. (eds.) Euro-Par 2018: Parallel Processing. Lecture Notes in Computer Science, vol. 11014, pp. 811–825. Springer (2018). https://doi.org/10.1007/978-3-319-96983-1_57

15. Charara, A., Keyes, D., Ltaief, H.: Batched Triangular Dense Linear Algebra Kernels for Very Small Matrix Sizes on GPUs. ACM Transactions on Mathematical Software 45(2) (2019). https://doi.org/10.1145/3267101

16. Corona, E., Martinsson, P.G., Zorin, D.: An $O(N)$ Direct Solver for Integral Equations on the Plane. Applied and Computational Harmonic Analysis 38(2), 284–317 (2015). https://doi.org/10.1016/j.acha.2014.04.002

17. Goreinov, S., Tyrtyshnikov, E., Yeremin, A.Y.: Matrix-Free Iterative Solution Strategies for Large Dense Linear Systems. Numerical Linear Algebra with Applications 4(4), 273–294 (1997)

18. Grasedyck, L., Kressner, D., Tobler, C.: A Literature Survey of Low-Rank Tensor Approximation Techniques. GAMM-Mitteilungen 36(1), 53–78 (2013). https://doi.org/10.1002/gamm.201310004

19. van Groenestijn, G.J., Verschuur, D.J.: Estimating Primaries by Sparse Inversion and Application to Near-offset Data Reconstruction. Geophysics 74(3), 1MJ–Z54 (2009). https://doi.org/10.1190/1.3111115

20. Hackbusch, W.: A Sparse Matrix Arithmetic Based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-Matrices. Computing 62(2), 89–108 (1999). https://doi.org/10.1007/s006070050015

21. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. SIAM Review 53(2), 217–288 (2011). https://doi.org/10.1137/090771806

22. Jumah, B., Herrmann, F.J.: Dimensionality-reduced Estimation of Primaries by Sparse Inversion. Geophysical Prospecting 62(5), 972–993 (2014). https://doi.org/10.1111/1365-2478.12113

23. Keyes, D.E., Ltaief, H., Turkiyyah, G.: Hierarchical Algorithms on Hierarchical Architectures. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 378(2166), 20190055 (2020). https://doi.org/10.1098/rsta.2019.0055

24. Kriemann, R.: H-LU Factorization on Many-Core Systems. Computing and Visualization in Science 16(3), 105–117 (2013). https://doi.org/10.1007/s00791-014-0226-7

25. Lindstrom, P.: Fixed-rate compressed floating-point arrays. IEEE Transactions on Visualization and Computer Graphics 20(12), 2674–2683 (2014). https://doi.org/10.1109/TVCG.2014.2346458

26. Ltaief, H., Cranney, J., Gratadour, D., *et al.*: Meeting the Real-Time Challenges of Ground-Based Telescopes Using Low-Rank Matrix Computations (2021), http://hdl.handle.net/10754/669813

27. van der Neut, J., Thorbecke, J., Wapenaar, K., Slob, E.: Inversion of the Multidimensional Marchenko Equation. In: 77th Conference and Exhibition, EAGE, Extended Abstracts. vol. 2015, pp. 1–5. European Association of Geoscientists & Engineers (2015). https://doi.org/10.3997/2214-4609.201412939

28. Ravasi, M., Vasconcelos, I.: PyLops – A Linear-operator Python Library for Scalable Algebra and Optimization. SoftwareX 11, 100361 (2020). https://doi.org/10.1016/j.softx.2019.100361

29. Ravasi, M., Vasconcelos, I.: An Open-source Framework for the Implementation of Large-scale Integral Operators with Flexible, Modern HPC Solutions - Enabling 3D Marchenko Imaging by Least Squares Inversion. Geophysics pp. 1–74 (2021). https://doi.org/10.1190/geo2020-0796.1

30. Ravasi, M., Vasconcelos, I., Kritski, A., *et al.*: Target-oriented Marchenko Imaging of a North Sea Field. Geophysical Journal International 205(1), 99–104 (2016). https://doi.org/10.1093/gji/ggv528

31. Rouet, F.H., Li, X.S., Ghysels, P., Napov, A.: A Distributed-memory Package for Dense Hierarchically Semi-separable Matrix Computations Using Randomization. ACM Transactions on Mathematical Software (TOMS) 42(4), 27 (2016). https://doi.org/10.1145/2930660

32. Verschuur, D.J.: Surface-related Multiple Elimination in Terms of Huygens Sources. Journal of Seismic Exploration 1, 49–59 (1992)

33. Wapenaar, K., Thorbecke, J., van der Neut, J., *et al.*: Marchenko Imaging. Geophysics 79(3), WA39–WA57 (2014). https://doi.org/10.1190/geo2013-0302.1

34. Williams, S., Waterman, A., Patterson, D.: Roofline: An Insightful Visual Performance Model for Multicore Architectures. Communications of the ACM 52(4), 65–76 (2009). https://doi.org/10.1145/1498765.1498785

35. Yilmaz, O.: Seismic Data Analysis. Society of Exploration Geophysicists (2001)