Please fill in the name of the event you are preparing this manuscript for.	MEOS 2019
Please fill in your 6-digit SPE manuscript number.	**SPE-194722-MS**
Please fill in your manuscript title.	Application of High Performance Asynchronous Acoustic Wave Equation Stencil Solver into a Land Survey

Please fill in your author name(s) and company affiliation.

Given Name	Surname	Company
Rached	Abdelkhalak	KAUST
Kadir	Akbudak	KAUST
Vincent	Etienne	SAUDI ARAMCO
Hatem	Ltaief	KAUST
Thierry	Tonellot	SAUDI ARAMCO
David	Keyes	KAUST

Abstract

This paper describes the application of high performance asynchronous stencil computations for 3D acoustic modeling on a synthetic land survey. Using the Finite-Difference Time-Domain (FDTD) method, a parallel Multicore Wavefront Diamond-tiling (MWD) stencil kernel (Malas et al. 2015, Malas et al. 2017) drives the high performance execution using temporal blocking to maximize data locality, while reducing the expensive horizontal data movement. As absorbing boundary conditions, we use Convolutional Perfectly Matched Layer (CPML), which have to be redesigned to not interrupt the asynchronous execution flow engendered by the MWD stencil kernel for the inner-domain points. The main idea consists in weakening the data dependencies by moving the CPML computations into the inner-computational loop of the MWD stencil kernel (Akbudak et al. 2019). In addition to handling the absorbing boundary conditions, applying the asynchronous MWD with CPML kernels to a realistic land survey requires the extraction of the wavefield value at each receiver position. We revisit the default extraction process and make it also compliant with the overall asynchrony of the 3D acoustic modeling. We report performance improvement up to 24% against the standard spatial blocking algorithm on Intel multicore chips using the synthetic land survey, which is representative of an area of interest in Saudi Arabia. While these results concur with previous performance campaign assessment, we can actually produce and assess the resulting 3D shot gather accuracy. To our knowledge, this is the first time the effectiveness of asynchronous MWD stencil kernel with CPML absorbing boundary conditions is demonstrated in an industrial seismic application.

Introduction

The Finite-Difference Time-Domain (FDTD) method has been used in geophysics for decades as the method of choice for seismic modeling. Its usage is nowadays intensive for applications that require accurate solution of the wave equation such as the reverse time migration (RTM) or full waveform inversion (FWI). Seismic modeling aims at computing synthetic data given an a-priori knowledge of the

sub-surface properties such as P-wave and S-wave velocities, anisotropic parameters, density, attenuation, just to cite a few. In the acoustic approximation, only P-wave propagates and if we assume the model is isotropic, then only a scalar P-wave velocity model is sufficient to perform seismic modeling. In the context of seismic imaging, such a model can be obtained with first arrival time tomography, for instance. In this work, a 3D synthetic P-wave isotropic velocity model is built with a geological stratigraphic forward modeling. This model is representative of an area of interest in Saudi Arabia. Our objective is to perform a synthetic seismic survey in this model using standard acquisition geometry. The computed data will eventually be used to assess the acquisition design and processing workflows (results not presented here).

Below, we investigate how spatial (Etienne et al. 2017) and temporal (Malas et al. 2015, Malas et al. 2017) cache blocking techniques can speed-up the FDTD computations on homogeneous multicores CPU x86 architectures. Both cache blocking techniques aim at increasing locally cached data, which is one of the most critical aspects for performance efficiency, especially when it comes to memory-bound algorithms like FDTD. However, temporal blocking technique creates further opportunities for data reuse along the time dimension. Based on parallel Multicore Wavefront Diamond tiling (MWD) (Malas et al. 2015, Malas et al. 2017), the stencil kernel computes the inner-domain points in an asynchronous fashion, while maximizing data reuse not only within a single core but also between cores sharing the same levels of cache. MWD enables to reuse freshly computed data solution between successive time steps within a diamond-shaped subdomain. Moreover, to treat the absorbing boundary conditions, we use Convolutional Perfectly Matched Layer (CPML) (Komatitsch and Martin 2007), which turns out to be challenging for MWD asynchronous execution. Indeed, CPML requires strong data dependencies at each time step, and therefore, necessitates breaking the asynchronous execution flow of MWD. As explained in Akbudak et al. (2019), we weaken these data dependencies by fusing the CPML computational loops with MWD kernel inner-loops at the expense of extra stencil operations. These extra operations may be reduced thanks to a buffer sliding window, while maintaining the overall asynchrony of the application and its performance superiority compared to spatial blocking techniques.

We assess the application of the asynchronous MWD with CPML kernels and check that previous reported performance results (Akbudak et al. 2019) are reproduced in the context of the herein seismic synthetic survey. One of the main challenges is to reduce the potential impact on performance when extracting the wavefield value at each receiver position. To overcome this challenge, the extraction phase has to be performed in an SPMD (single program multiple data) fashion by distributing the receivers' array across diamond-shaped domains. The obtained performance results show that we are able to sustain up to 24% performance improvement against standard spatial blocking for the 3D acoustic modeling on the latest Intel multicore chip. To our knowledge, this is the first time the effectiveness of the MWD stencil kernel with CPML absorbing conditions is demonstrated in an industrial seismic application.

The remainder of the paper is structured as follows. We present the main characteristic of the seismic modeling with the FDTD. The spatial blocking and temporal blocking algorithms are detailed in two distinct sections. An application to a 3D land model allows assessing the performance of these algorithms on the supercomputer Shaheen II from the King Abdullah University of Science and Technology (KAUST). To conclude, we summarize the benefits of our approach for seismic imaging and inversion and highlight future works.

Seismic modeling with the Finite-Difference Time-Domain method

FDTD relies on the evaluation of the wave equation on grid points generally distributed on a Cartesian grid. In this study, we consider the second-order acoustic wave equation in a 3D medium with constant density

$$\frac{\partial^2 p}{\partial t^2} = c^2\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2}\right), \quad (1)$$

where p is the pressure wavefield (space and time dependent) and c the wave velocity (only space dependent). To solve this equation, we adopt a second-order FD operator in time and an Nth-order FD operator in space to evaluate the partial derivatives. The corresponding discrete system can be expressed as

$$\frac{P_{ijk}^{n+1} - 2P_{ijk}^{n} + P_{ijk}^{n-1}}{\Delta t^2} = c_{ijk}^2\left(O_{ii}^N(P_{ijk}^n) + O_{jj}^N(P_{ijk}^n) + O_{kk}^N(P_{ijk}^n)\right), \quad (2)$$

where P_{ijk}^n is the value of the pressure at time step n at grid point with indexes ijk, Δt is the time step, and O_{ii}^N denotes the Nth-order FD spatial operator to evaluate the spatial second derivative along index i. The accuracy of the numerical scheme depends on the considered stencil, defined by the number of grid points used to approximate the spatial derivatives. We select the 8th-order FD spatial operator because it offers a high accuracy and a reasonable computational cost. A discretization of four points per wavelength is sufficient to achieve the required accuracy for our application. Such scheme requires eight neighboring grid points to the central point along each Cartesian direction, as depicted in Figure 1.

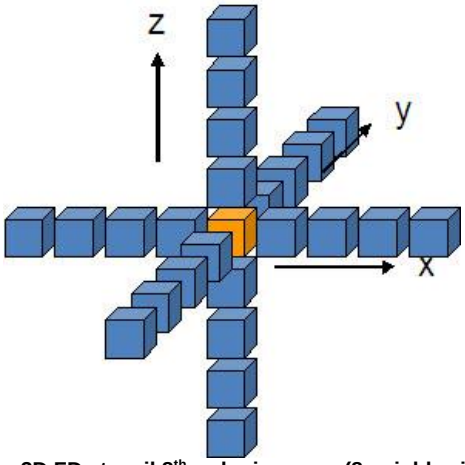

Figure 1 – Schematic view of the 3D FD stencil 8th order in space (8 neighboring points to the central points)

From an algorithmic point of view, the implementation of FDTD requires storage of three 3D-arrays: two arrays for the pressure at two distinct time steps and one array for the wave velocity. It is not necessary to keep all the time samples since only the current state of the pressure P_{ijk}^n and the previous one P_{ijk}^{n-1} are required to compute the next time step P_{ijk}^{n+1}. The computation workload is related to the number of floating-point operations (i.e., flops, such as add and multiply) and the number of memory operations (load and store data in memory) required in Equation (2). For the 8th-order in space, there are 48 flops for each grid point at each time step, as indicated in Table 1.

At the edges of the computational domain, a boundary condition needs to be implemented in order to absorb artificial reflections. We adopt the Convolutional Perfectly Matched Layer (CPML) (Komatitsch and Martin 2007). Six layers (one for each edge of the 3D model) with 10-point width allow to efficiently

damp the acoustic wavefield. CPML requires modifying the wave equation by introducing new variables. Consequently, the flops are augmented by 50 for each grid point in the CPML compared to the inner domain, as shown in Table 1.

Spatial blocking

A simple FDTD implementation would sequentially access all grid points via three nested loops over indices i, j, k and perform computations following Equation (2). To enable parallelism on multicore architectures, one may include OpenMP directives to parallelize these loops and take advantage of the shared-memory. On such architectures, it is beneficial to implement cache blocking to work around memory bandwidth limitations and to increase performance. The concept of spatial blocking (SB), illustrated in Figure 2, is to divide the computational grid into subdomains that can fit into the CPU cache (the lowest-level cache L3 is often sufficient when it is large enough) for efficient memory cache reuse. Each OpenMP thread is associated with a subdomain characterized by its size along the three directions. Schematically, the algorithm contains three nested loops parallelized with OpenMP over the blocks, and three nested loops over the grid points i, j, k of the block. In our implementation, the inner index (memory is contiguous along this index) is i, followed by indices j and k. In the C++ language, P^n_{ijk} is then translated into P[k][j][i]. To provide a map between the computing indices and the spatial dimensions, we adopt the following convention: index i (fast index) corresponds to the z-axis, index j to the y-axis, and index k (slow index) to the x-axis.

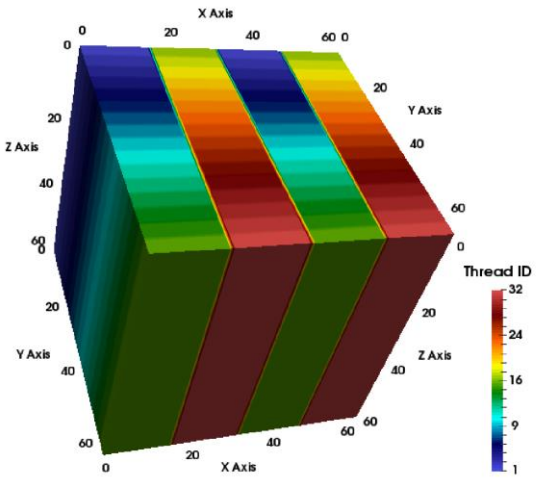

Figure 2 – Distribution of 32 OpenMP threads within a 64 x 64 x 64 3D FDTD grid following a cache-blocking algorithm. The continuous dimension in memory (z-axis) is left unchanged to keep efficient SIMD vectorization. The x and y dimensions are partitioned into smaller domains (16 x 4) for efficient memory cache reuse during FDTD computation.

Finding the optimal block size is not straightforward since it depends on the FD scheme stencil, the model size and the underlying hardware (e.g., number of threads and cache size). We follow a simple approach proposed by Etienne et al. (2017), to find the optimal setting. We perform a series of computations where the block size is parameterized and different for each computation. We realize 1024 tests to measure the performance when the block size varies from 1 to 32 points, both in *x* and *y*. We assume it is better not to divide the domain along *z*, which corresponds to the inner index in order to keep an efficient hardware vectorization through Intel AVX instructions. We perform the test using a 512x512x512 grid points without CPML and determine that the optimal block size is 16 points along x, and 5 points along y (Etienne et al. 2017). These results are obtained using 32 OpenMP threads on one computing node of the supercomputer Shaheen II at KAUST, based on a dual-socket 16-core Intel Haswell CPU.

Temporal blocking

For temporal blocking (TB), we use the parallel multicore wavefront diamond tiling (MWD) with CPML. In addition to spatial blocking, MWD provides an extra dimension for data reuse along the time axis. The asynchronous time integration renders the 3D main domain into diamond-shaped subdomains of multiple time steps' height. Therefore, MWD further increases the vertical data motion within the high level of the memory subsystem, while releasing memory bandwidth pressure. The multicore wavefront mechanism enables a tightly coupled group of threads to compute the data solution within each diamond, while taking advantage of date reuse among shared caches. We refer the reader to (Malas et al. 2015, Malas et al. 2017) for further details on MWD. Unfortunately, the CPML formulation in terms of data dependencies prevents the MWD stencil kernel from running in full asynchronous mode of execution. Indeed, the default CPML implementation requires global synchronization between threads at each time step. The main idea of (Akbudak et al. 2019) is to restore the asynchronous behavior of MWD in presence of CPML by fusing their respective inner computational loops. However, this loop fusion has dramatic performance penalty, as it introduces extra floating-point operations. To cut down these extra operations, a sliding window buffer is employed to store transient data solutions and to maintain the original integrity of the asynchronous execution flow of the MWD stencil kernel.

Figure 3 shows how the MWD thread partitioning operates in the x and y dimensions, along the time axis. Besides being the fastest index, we omit the z dimension for simplicity purposes. The y dimension corresponds to the dimension along which diamond-shaped domain decomposition occurs. The diamond shape is the result of data dependency across time: if at time step *t* the domain ranging from y_{begin} to y_{end} has been updated, we can proceed updating the domain ranging from y_{begin}+*half_stencil_width* to y_{end}-half_stencil_width without synchronization. The x dimension is the dimension where the multicore parallelism runs within the wavefront.

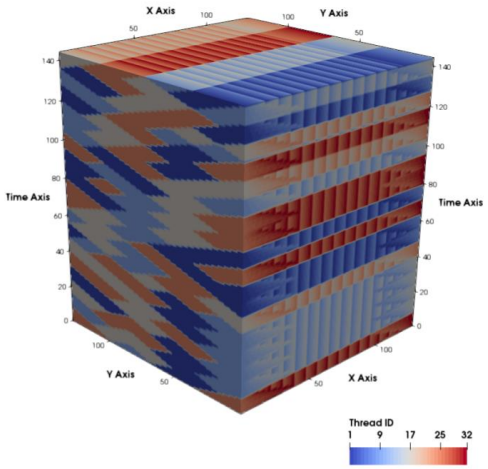

Figure 3 – MWD thread partitioning in X and Y dimensions, along the time axis.

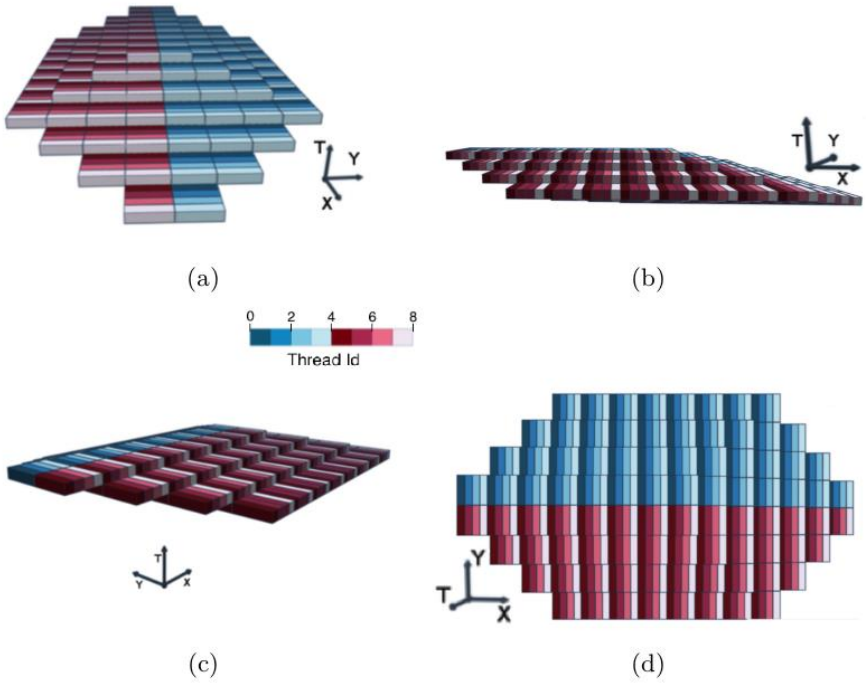

Figure 4 – Parallel wavefront mechanism in MWD.

Figure 4 shows in more details the internal of the parallel wavefront mechanism in MWD, which is paramount for achieving high performance. The idea is similar to the diamond concept in y/t plan: several time steps are executed within a given wavefront size before moving to the next wavefront in the x direction. Figures 4(a-d) present a single diamond-shaped domain with domain decomposition along the y dimension while Figures 4(b-c) highlight the wavefront parallelism along the x dimension (gradient colors).

Once applied to the seismic synthetic survey, it is critical to tune the parameters of the MWD with CPML kernels to achieve high performance execution. There are three main tuning parameters. The first one corresponds to the number of time steps inside the diamonds (TDIM), which creates opportunities for single-core data reuse within diamond-shaped domains. The second parameter defines the two-level OpenMP-based nested parallelism, i.e., the number of thread groups (NTG) operating on all independent diamonds (inter-diamond parallelism) as well as the number of threads (NTPG) operating within a single diamond (intra-diamond parallelism) and along each direction (NTHX, NTHY and NTHZ). The third parameter is the wavefront size (NWF), which creates opportunities for data reuse among multicores within diamond-shaped subdomains. The detailed optimization study of these three intertwined parameters is more challenging than SB. We employ a basic brute force approach based on an "*onion peeling"* strategy to reduce the tuning time and retrieve a close to optimal configuration leading to decent performance. The brute force approach creates a database of best configurations for each problem size and picks the closest one to the considered problem at runtime. We have also considered an auto-tuning mechanism to explore different configurations for a limited number of time steps and have selected the best one. A more thorough tuning study may be important for future work and is out of the paper scope.
Application of MWD with CPML kernels to a synthetic seismic survey requires the extraction of the wavefield value at each receiver position. This critical step is straightforward for the spatial blocking approach due to the bulk synchronous programming model nature of the kernel. Once the CPML contributions are applied to the boundary data solutions, the entire array containing the indices of the

receivers may be traversed only once during an atomic phase of extraction. This is not applicable for the asynchronous MWD kernel, since each diamond has a limited extent across y spatial dimension. This means that every diamond has to read the entire array containing receivers' indices, which impedes parallel performance. This phenomenon is further exacerbated due to intensive memory accesses, in case there are thousands of receivers. To overcome this challenge, the extraction phase has to be performed in an SPMD (single program multiple data) fashion. This enables each diamond to consider only its corresponding small subset of all receivers when saving traces, which drastically reduces the observed performance penalty.

Application to a land survey

Model and acquisition geometry

Velocity and density models are built with a geological stratigraphic forward modeling. These models are representative of an area of interest in Saudi Arabia. Although, P-wave velocity, S-wave velocity and density models were created, only the P-wave velocity model is used in this work. The model dimension is 17 km (inline) x 14.3 km (crossline) x 5.2 km (depth).

A regular acquisition geometry with a maximum offset of 5 km inline and crossline is considered. Hence, the receiver coverage area is 10 km x 10 km. To perform the modeling, an extraction of the model covering this area was done (Figure 5). The grid size is 1001 x 1001 x 849 points in x, y, and z, respectively. The grid spacing is 10 m horizontally and 6.1 m vertically. A dense acquisition with 50 m spacing between receivers per receiver line and 100 m between receiver lines is considered. This acquisition allows subsequent decimation of the data in order to assess different acquisition scenarios. The source function exhibits a maximum frequency of 50 Hz. Total modeling time is 3 s and the time step is 0.5 ms (i.e. total number of time steps in FDTD is 6000). Boundary absorbing CPML 10-point width is applied at edges of the computational domain.

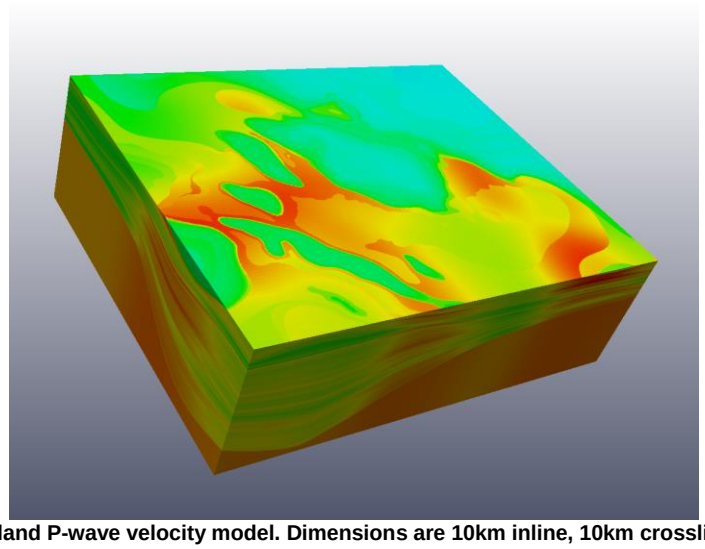

Figure 5 – Synthetic land P-wave velocity model. Dimensions are 10km inline, 10km crossline and 5.2 km depth.

Numerical results

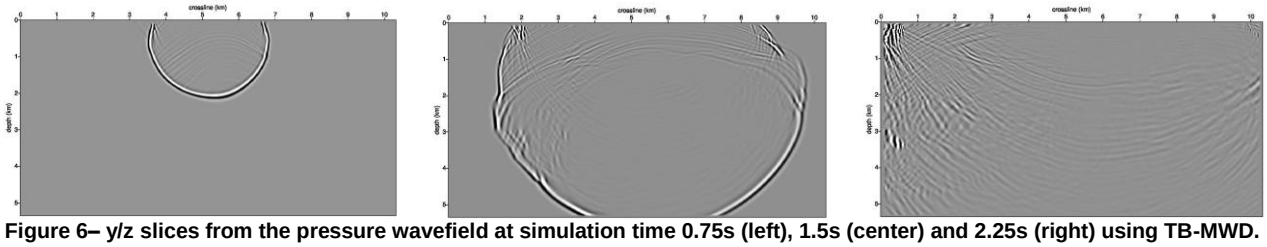

Figure 6– y/z slices from the pressure wavefield at simulation time 0.75s (left), 1.5s (center) and 2.25s (right) using TB-MWD.

Figure 6 shows three slices extracted from the 3D pressure wavefield along y/z plan at the source x position for different time iterations corresponding to the reported simulation times. They show no numerical dispersion during the wave propagation. No reflections are observed at the domain boundaries, which illustrates the effectiveness of the 10-point width CPML.

Figure 7 shows three sets of traces corresponding to three different receiver lines computed with TB-MWD extracted from the same 3D shot point gather. Gathers are coherent with the velocity model in use. The snapshots and the shot point gathers obtained with the TB-MWD implementation are compared to the reference SB implementation and show acceptable RMS deviation (<1E-3). All these elements prove the numerical robustness of the TB-MWD approach.

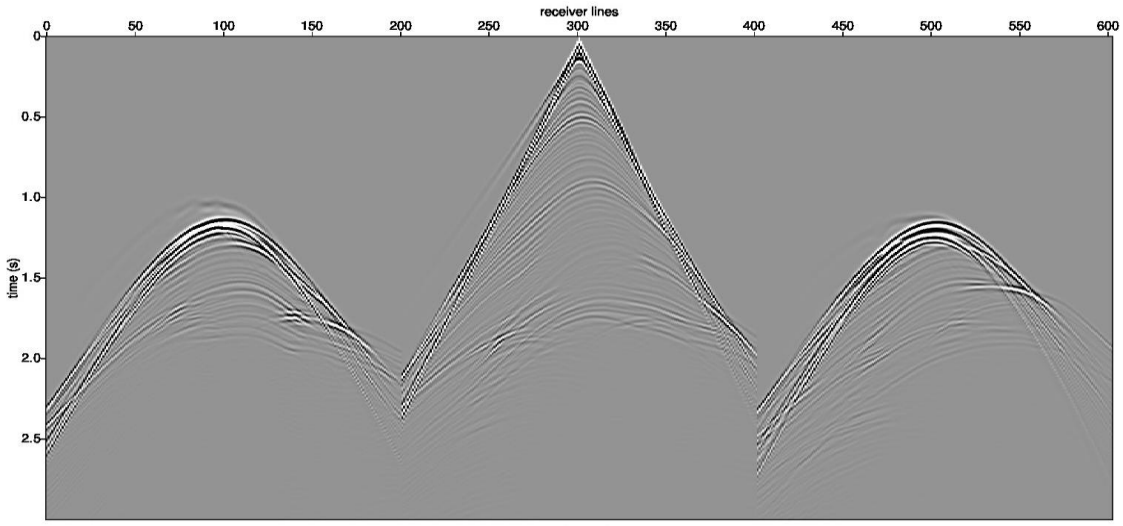

Figure 7 – Three receiver lines extracted from the 3D shot gather.

Performance analysis

Table 1 and 2 summarize the measured performance when the seismic modeling is performed with the FDTD algorithm with spatial blocking (SB) and TB-MWD techniques on two different platforms. First platform is a dual-socket 16-core Intel Haswell E5-2698v3 CPU with AVX-2 running at 2.30GHz. The L3 cache size is 40MB. The RAM size is 128 GB per node and the sustained memory bandwidth is 111 GB/s. The second platform is a dual-socket 28-core Intel Skylake Platinum 8176 CPU with AVX-512

running at 2.10GHz. The L3 cache size is 38.5MB. The RAM size is 192 GB per node and the sustained memory bandwidth is 182 GB/s. For both platforms Intel compiler icc v18.0.1 is used.

Haswell platform

Technique			SB				TB-MWD			
Phase		#Flop/point	Time (s)	Time (%)	Gpoint/s	GFlops/s	Time (s)	Time (%)	Gpoint/s	GFlops/s
Initialization		-	7,9	0,7	-	-	8,5	0,8	-	-
Time iterations	Inner domain	48	892,9	75	6,09	292	599	54,5	9,07	435,55
	CPML x	50	59,4	5,0	1,8	90	200	18,2	0,53	26,62
	CPML y	50	48,6	4,1	2,2	110	160	14,6	0,67	33,27
	CPML z	50	165,1	13,9	0,8	38	130	11,8	0,96	48,11
	Disk IO	-	16,1	1,4	-	-	0,9	0,1	-	-
Total		**-**	**1190**	**100**	**-**	**-**	**1098,4**	**100,0**	**-**	**-**

Table 1 – Statistics of the run with the spatial blocking and temporal blocking algorithms on one computing node of Shaheen (Intel Haswell).

The performance results on the Haswell platform are reported in Table 1, using the following TB-MWD parameter configuration: TDIM=7, NTG=4, NTPG=8, NTHX=8, NTHY=1, NTHZ=1, NWF=16. For the SB algorithm, the total computation time is 1190s, with 75% and 23% of the elapsed time spent in the inner kernel and in the CPML, respectively. Initializations (memory allocation, reading velocity model from disk) and output disk operations (i.e., writing seismic traces) represent only 2% of the run time. The discrepancy between the computational speed of the inner kernel and the CPML (about 300 Gflops/s for inner kernel and from 38 to 110 Gflops/s in the CPML) is striking. In the worst case (top and bottom absorbing layers), the rate of execution is almost one order of magnitude lower. This explains the fact that although the CPML regions represent only 4% of the total grid points, the CPML computations take nearly a quarter of the total time. The main cause of the poor performance in the absorbing layers is the lack of vectorization. This is especially critical for the top and bottom CPMLs, where only 10 points are found along the z-axis (along which memory access is continuous), which prevents efficient vectorization.

For the TB-MWD algorithm, the overall timing is 1098s, which represents a speedup of 8% compared to the SB algorithm. The CPML impact is even more striking than in the SB case: 44.6% of time is induced by the CPML overhead. Even though it avoids redundant computations in the CPML regions, the introduced sliding window buffer for TB-MWD (Akbudak et al. 2019) may explain the observed performance penalty. Indeed, there is a strided memory access along the x and y dimensions larger than the z dimension, which makes the CPML performance significantly drop for the former dimensions. However, when considering the inner domain, the performance comparison clearly favors the TB-MWD

implementation. The speedup for the inner domain is close to 50%. Furthermore, for larger models where CPMLs and their impact are reduced in proportion, higher overall speedups can be expected. Indeed, performance results reported in Akbudak et al. (2019) indicate that a 30% speedup can be achieved for larger models on the similar platform.

While the initialization time is close to the SB case, the time to write the snapshot is significantly reduced at the expense of a larger memory consumption. In fact, the 3D gathers are kept in memory for all the time iterations whereas they are saved to disk as they are generated in SB. The asynchronous nature of TB-MWD involves that the time iteration reached by grid points is not consistent across the whole wavefield. The same strategy for dumping traces can also be applied to SB, resulting in the same effect.

Skylake platform

Technique			SB				TB-MWD			
Phase		#Flop/point	Time (s)	Time (%)	Gpoint/s	GFlops/s	Time (s)	Time (%)	Gpoint/s	GFlops/s
Initialization		-	0.4	0.1	-	-	2,11	0,3	-	-
Time iterations	Inner domain	48	613.5	73.9	8.86	425.32	445,5	66,9	12,20	585,62
	CPML x	50	55.5	6.7	1.93	96.67	103,5	15,5	1,03	51,43
	CPML y	50	37.0	4.5	2.87	143.5	57	8,6	1,87	93,39
	CPML z	50	114.6	13.8	1.09	54.5	57	8,6	2,19	109,73
	Disk IO	-	8.7	1.0	-	-	0,9	0,1	-	-
Total		**-**	**829.7**	**100.0**	**-**	**-**	**666,01**	**100,0**	**-**	**-**

Table 2 – Statistics of the run with the Spatial Blocking and Temporal Blocking algorithms on one Intel Skylake node

Table 2 highlights the performance results on the Skylake platform, with the following TB-MWD parameter configuration: and TDIM=7, NTG =4, NTPG=14, NTHX=7, NTHY=2, NTHZ=1, NWF=28. They show that, for the SB algorithm, the total computation time is reduced to 830s. However, the inner domain and CPML computation time proportions remain approximately the same as for the Haswell platform and the same reasoning applies also here (i.e. vectorization issues).

For the TB-MWD algorithm, the overall timing is reduced to 666s, which represents a speedup of 24% compared to the SB algorithm. The effect of the higher memory bandwidth and intra-diamond parallelism on Skylake favorably impacts our CPML implementation, reducing the CPML time proportion from 44.6% to 32%. The speedup for the inner domain is close to 40%. Results reported in Akbudak et al. (2019) indicate that a 33% speedup can be achieved for larger models on the similar platform.

Toward leveraging the performance of the Reverse Time Migration

This section details some necessary steps toward leveraging the performance of the reverse time migration. Figure 8 pictures the performance degradation when forcing MWD to interrupt its asynchronous execution across all diamonds so that atomic I/O operations can occur such as wavefield snapshotting at certain time step intervals. If the frequency is high, MWD faces significant performance degradation compared to SB and even become slower. On the opposite, if the frequency is low (i.e., from a frequency of 58 time steps onward), MWD does not get affected and may pursue the time integration with the proper performance regime. But this is not an option since snapshotting every 58 time steps is not suitable for RTM which requires higher frequency for snapshotting.

To alleviate this difficulty, we envision to redesign the implementation of the I/O phase during the forward (i.e., write) and adjoint (i.e., read) phase of RTM, and ultimately, the calculation of the image condition. The idea is to use fine-grained workload decompositions for both phases, i.e., the I/O operations and the computation of the image condition. This workload decomposition for I/O and the image condition have to match the granularity of the diamond-shaped domain, while adopting the asynchronous execution flow of MWD. A careful housekeeping needs to be done to ensure the image condition is computed only between coherent fine-grained snapshots. Moreover, while the granularity should be identical for the diamond-shaped domain and the I/O snaphshots, the image condition does not require having all diamond-shaped domain to be stored at the same time step. This is the case for spatial blocking though, but this is rather due to the bulk synchronous programming model employed within its implementation. The only requirement for getting the proper accuracy is to ensure a one-to-one fine-grained coherency at the time synchronization point, just before entering the fine-grained image condition, between the *I/O* time step during the forward (i.e., write) / adjoint (i.e., read) time integration and the *compute* time step of the imaging condition. This inherent time stepping flexibility of the image condition is not taken into account for the spatial blocking. However, it may be leveraged for asynchronous MWD. Indeed, it may permit to reduce pressure and prevent saturation on various critical hardware resources, by intermittently soliciting them.

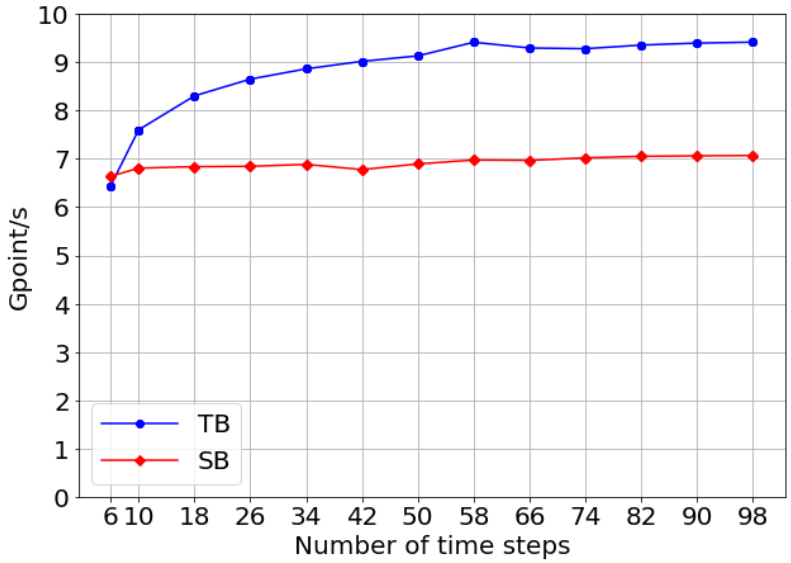

Figure 8 – Performance impact on SB/MWD depending on the I/O snapshotting frequency.

Conclusion and future work

This paper presents the application of high performance asynchronous MWD with CPML kernels into a realistic land survey, which is representative of an area of interest in Saudi Arabia. To our knowledge, this is the first time asynchronous MWD kernels with CPML absorbing boundary conditions drive an industrial seismic application. The reported results present up to 24% performance speedup against the standard SB stencil kernels. Beyond pure seismic modeling applications, there are still some ongoing efforts to fully apply asynchronous MWD to RTM, including fine-grained I/O snapshotting. Another future work is to port these stencil kernels on GPUs and assess the performance impact of data movement across the slow PCIe bus as well as between GPUs via the fast NVLink interconnect.

Acknowledgement

Computations were done on KAUST's Shaheen II supercomputer. We acknowledge the support of the KSL supercomputing lab.

References

Akbudak, K., Ltaief, H., Etienne, V., Abdelkhalak, R., Tonellot, T., Keyes, D.: Ubiquitous Asynchronous Computations for Solving the Acoustic Wave Propagation Equation. http://hdl.handle.net/10754/630323, submitted to International Conference on Supercomputing (2019)

Etienne, V., Tonellot, T., Thierry, P., Berthoumieux, V., & Andreolli, C.: Optimization of the seismic modeling with the time-domain finite-difference method. In *SEG Technical Program Expanded Abstracts 2014* (pp. 3536-3540). Society of Exploration Geophysicists (2014)

Etienne, V., Tonellot, T., Malas, T., Ltaief, H., Kortas, S., Thierry, P., Keyes, D.: High-Performance Seismic Modeling with Finite-Difference Using Spatial and Temporal Cache Blocking. 3rd EAGE Workshop High Performance Computing for Upstream (2017)

D. Komatitsch and R. Martin. An Unsplit Convolutional Perfectly Matched Layer Improved at Grazing Incidence for the Seismic Wave Equation. GEOPHYSICS, 72(5):SM155–SM167, 2007.

Malas, T., Hager, G., Ltaief, H., Stengel, H., Wellein, G., Keyes, D.: Multicore Optimized Wavefront Diamond Blocking for Optimizing Stencil Updates. SIAM Journal on Scientific Computing 37(4), 439–464 (2015)

Malas, T., Hager, G., Ltaief, H., Keyes, D.: Multidimensional Intratile Parallelization for Memory-Starved Stencil Computations. ACM Trans. Parallel Comput. 4(3), 12:1–12:32 (2017)