Performance Evaluation of Pseudospectral Ultrasound Simulations on a Cluster of Xeon Phi Accelerators

Filip Vaverka[1][0000−0002−7960−5752], Bradley E. Treeby[2][0000−0001−7782−011X], and Jiri Jaros[1][0000−0002−0087−8804]

[1] Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 2, 612 00 Brno, Czech Republic
{ivaverka,jarosjir}@fit.vutbr.cz

[2] University College London, Medical Physics and Biomedical Engineering, WC1E 6BT London, United Kingdom
b.treeby@ucl.ac.uk

Abstract. The rapid development of novel procedures in medical ultrasonics, including treatment planning in therapeutic ultrasound and image reconstruction in photoacoustic tomography, leads to increasing demand for large-scale ultrasound simulations. However, routine execution of such simulations using traditional methods, e.g., finite difference time domain, is expensive and often considered intractable due to the computational and memory requirements. The k-space corrected pseudospectral time domain method used by the k-Wave toolbox allows for significant reductions in spatial and temporal grid resolution. These improvements are achieved at the cost of all-to-all communication, which are inherent to the multi-dimensional fast Fourier transforms. To improve data locality, reduce communication and allow efficient use of accelerators, we recently implemented a domain decomposition technique based on a local Fourier basis.

In this paper, we investigate whether it is feasible to run the distributed k-Wave implementation on the Salomon cluster equipped with 864 Intel Xeon Phi (Knight's Corner) accelerators. The results show the immaturity of the KNC platform with issues ranging from limited support of Infiniband and LustreFS in Intel MPI on this platform to poor performance of 3D FFTs achieved by Intel MKL on the KNC architecture. Yet, we show that it is possible to achieve strong and weak scaling comparable to CPU-only platforms albeit with the runtime 1.8× to 4.3× longer. However, the accounting policy for Salomon's accelerators is far more favorable and thus their employment reduces the computational cost significantly.

Keywords: Ultrasound simulations · Local Fourier basis decomposition · Pseudospectral methods · Ultrasound · k-Wave toolbox · Intel Xeon Phi · Knight's Corner · MKL · MPI · OpenMP.

1 Introduction

There is a growing number of medical applications of ultrasound such as photoacoustic imaging [1], neurostimulation [28] and high intensity focused ultrasound (HIFU) cancer treatment [5,18]. These applications require fast, accurate and versatile ultrasound propagation models in tissue-like materials at various stages such as planning or post-processing. Typically, these requirements can be met with models based on the generalized Westervelt equation [22], which allows for modeling nonlinear ultrasound wave propagation through heterogeneous medium with a power law absorption. Due to this demand, several ultrasound modeling packages for medical applications have been released along with our k-Wave toolkit, see [9] for a recent review. The majority of those packages employ either finite-difference time-domain (FDTD) method, pseudospectral time-domain (PSTD) method or a variant of operator-splitting methods. The k-Wave toolbox is among a few based on the k-space pseudo-spectral time-domain (KSTD) method.

FDTD methods scale well on large parallel systems using a straightforward domain decomposition and halo exchange over the nearest neighbors [31]. However, most FDTD schemes require between 8 to 10 grid points per wavelength to achieve sufficient accuracy and even more to manage dispersion in cases where propagation over a large number of wavelengths has to be modeled [17]. This makes many realistic ultrasound simulations intractable due to high memory requirements. The PSTD methods can theoretically approach the Nyquist limit of 2 grid points per wavelength and thus significantly reduce the memory requirements. The KSTD method improves on the PSTD method by using a semi-analytical time-stepping schemes [25] which complements excellent spatial properties with a larger time step size. The main drawback of PSTD and especially KSTD methods is the introduction of a global trigonometric basis and the use of the fast Fourier transform (FFT) to implement gradient operators. The KSTD method requires, due to the k-space correction, 3D FFTs which inherently limit the scalability of this method on parallel distributed, and especially on accelerated systems [13]. Although a lot of work on efficient distributed FFTs has been carried out (FFTW [7], Hybrid FFTW [20], P3DFFT [21], PFFT [23], AccFFT [8], multi-GPU CUDA FFT [19] or FFT-ECP [26]), the computation time is still often determined by the communication between subdomains, which in many cases prevents the use of accelerators such as GPUs or Intel Xeon Phis.

This paper investigates the possibility of deploying a distributed implementation of the KSTD method implemented in the k-Wave toolbox [14] on a large cluster of Intel Xeon Phi accelerators. The method combines advantages of FDTD and KSTD methods by replacing global Fourier basis with a set of local ones [12], thus achieving communication complexity of an FDTD method while maintaining many properties of a KSTD method. The following section briefly describes the principle of the local Fourier basis decomposition. Next, the architecture of the accelerated cluster is described in Sec 3. After that, the implementation is briefly explained in Sec 4 and achieved scaling results are presented in Sec 5.

Finally, Sec 6 investigates performance of the accelerators and the last section draws conclusions on usability of the platform.

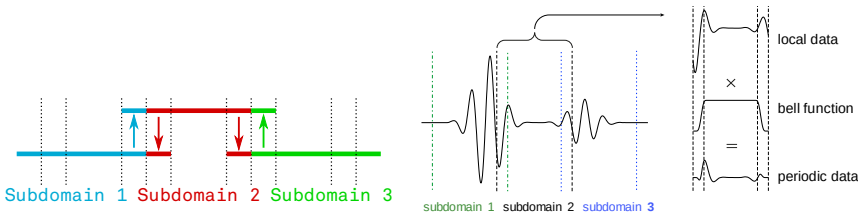
2 Local Fourier Basis Domain Decomposition

The numerical model of the nonlinear wave propagation in heterogeneous absorbing medium investigated in this paper is based on the governing equations derived by Treeby [27] written as three-coupled first-order partial differential equations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= -\frac{1}{\rho_0}\nabla p + \mathbf{F} \ , && \text{(momentum conservation)} \\ \frac{\partial \rho}{\partial t} &= -\rho_0 \nabla \cdot \mathbf{u} - \mathbf{u} \cdot \nabla \rho_0 - 2\rho \nabla \cdot \mathbf{u} + M \ , && \text{(mass conservation)} \\ p &= c_0^2 \left(\rho + \mathbf{d} \cdot \nabla \rho_0 + \frac{B}{2A}\frac{\rho^2}{\rho_0} - \mathrm{L}\rho \right) . && \text{(equation of state)} \end{aligned} \tag{1}$$

Here $\mathbf{u}$ is the acoustic particle velocity, $\mathbf{d}$ is the acoustic particle displacement, p is the acoustic pressure, ρ is the acoustic density, ρ_0 is the ambient (or equilibrium) density, c_0 is the isotropic sound speed, B/A is the nonlinearity parameter, and operator L captures power law absorption and dispersion. Two linear source terms are also included, where $\mathbf{F}$ is a force source term, and M is a mass source.

In the k-Wave toolbox, the model (1) would normally be directly discretized using the KSTD method (see [27]). However, to alleviate the global communication overhead, we instead divide the simulation domain into a set of cuboid subdomains, each of which supported by its own local Fourier basis [12] (LFB for short). The neighboring subdomains are coupled by overlapping each other and the overlaps are also used to restore their local periodicity [2], see Fig. 1.



(a) Overlap exchange (b) Periodicity restoration

Fig. 1: The principle of local Fourier basis domain decomposition shown for one spatial dimension. (a) The local subdomain is padded with a few grid points (overlap) from both neighboring subdomains which are periodically exchanged. (b) After the overlap exchange, each local domain is multiplied by a bell function to restore periodicity.

The computation itself consists of an iterative algorithm running over a given number of time steps. Each time step is composed of a sequence of element-wise operations, overlap exchanges and local 3D FFTs, see Fig. 2. The majority of the computation time is usually spent on 3D FFTs or overlap exchanges. The restriction of the Fourier basis to the local subdomain has naturally a negative impact on the accuracy of the LFB method. The amount of accuracy loss depends on the overlap size and the properties of the bell function used [3,14,29].

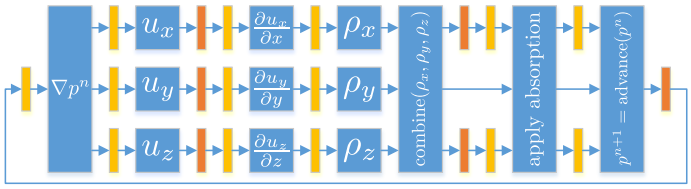


Fig. 2: Simplified computation loop governed by Eq. (1). Blue blocks denote element-wise operations, yellow 3D FFTs, and orange overlap exchanges.

3 Target Architecture

The target architecture investigated in this paper is a typical Intel/Infiniband cluster accelerated with Intel Xeon Phi cards of the Knight's Corner architecture [15]. The experiments were conducted on the Salomon supercomputer at the IT4Innovations national supercomputing center in Ostrava, Czech Republic[3].

Salomon consists of 1008 compute nodes, 432 of which are accelerated by Intel Xeon Phi 7120P accelerators. The architecture of the Salomon's accelerated part is shown in Fig. 3. Every node consists of a dual socket motherboard populated with two Intel Xeon E5-2680v3 (Haswell) processors accompanied with 128 GB of RAM. The nodes also integrate a pair of accelerators connected to individual processor sockets via the PCI-Express 2.0 x16 interface. The communication between processors is handled by the Intel QPI interface.

The nodes are interconnected by a 7D enhanced hypercube running on the 56 Gbit/s FDR Infiniband technology. The accelerated nodes occupy a subset of this topology constituting a 6D hypercube. Every node contains a single Infiniband network interface (NIC) connected via PCI-Express 3.0 to the first socket, and a service 1 Gbit/s Ethernet interface connected to the same socket. Both accelerators are capable of directly accessing the Infiniband NIC by means of Remote Direct Memory Access (RDMA).

A single Intel Xeon Phi 7120P accelerator packs 61 P54C in-order cores extended by 4-wide simultaneous multithreading (SMT) and a 512-bit wide vector

[3] `https://docs.it4i.cz/salomon/hardware-overview/`

processing unit (VPU). The Xeon Phi cores are supported by 30.5 MB of L2 cache evenly distributed over individual cores and interconnected via a ring bus. The memory subsystem consists of 4 memory controllers managing in total 16 GB of GDDR5. The theoretical performance and memory bandwidth of a single accelerator is over 2 TFLOP/s in single precision and 352 GB/s, respectively. A single accelerator can theoretically provide a speedup of 4× for compute bound, and 5× for memory bandwidth bound applications over a single twelve core Haswell processor. The total compute power of the accelerated part of the cluster reaches one PFLOP/s.

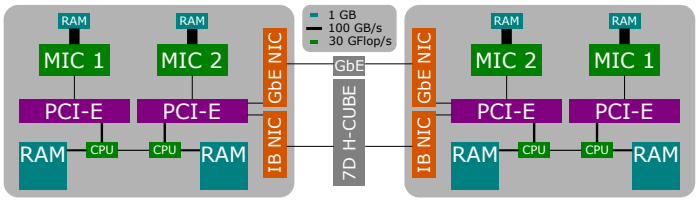


Fig. 3: The architecture of the Salomon accelerated nodes and interconnection. The size of the rectangles representing individual components is proportional to their performance, bandwidth or capacity.

4 Implementation

4.1 Execution Mode

The Intel Xeon Phi offers three different modes of execution: (1) The offload mode is an analogy to the GPGPU approach where the accelerator is controlled by the CPU and used only for the compute intensive tasks. (2) The native mode uses the accelerator as an isolated compute node with shared memory. (3) The cluster mode allows accelerators to be connected using the message passing interface (MPI) and run distributed jobs over many accelerators. In this mode, the CPUs can be used either to only run the operating system, or join the distributed job as additional workers, although with reduced compute power.

The proposed implementation uses Intel Xeon Phis in the native mode, which allows for direct access to NICs through the RDMA mechanism and thus should achieve the best performance in MIC-to-MIC communication. Overall the PSTD and KSTD methods tend to be memory bound as they exhibit relatively low arithmetic intensity on the order of $O(\log n)$. Therefore, the k-Wave toolbox favors architectures with high memory bandwidth and fast access to the network.

The code is logically structured into one MPI process per subdomain which runs on a single accelerator (or CPU socket). The work local to each subdomain is distributed across cores by means of OpenMP and OpenMP SIMD constructs. Since realistic simulations do not require double precision, only single precision

floating point operations are used. This yields higher performance and saves valuable memory bandwidth. The k-Wave LFB code boils down to a mix of element-wise operations on 3D real or complex matrices, 3D fast Fourier transforms and overlap exchanges. The following sections briefly describe most important operations, however, more details can be found in [29].

4.2 Fast Fourier Transforms

The most computationally expensive part of the simulation loop consists of a number of 3D fast Fourier transforms calculated over the local subdomains. Depending on which medium parameters are heterogeneous, the number of FFTs varies between ten and fourteen. Their actual implementation relies on third party libraries compatible with the FFTW interface [7], in this case the Intel MKL library [11] which showed better results than FFTW on the Intel Xeon Phi architecture [30].

The simulation code mostly uses out-of-place real-to-complex and complex-to-real transforms which reduces both the spatial and temporal complexity of the FFT by a factor of two [24]. However, the implementation of the out-of-place C2R transforms in the MKL library has proved to be very inefficient on the Intel Xeon Phi Knight's Corner. Hence, the C2R transforms are performed in-place using a temporary matrix and the results consequently copied to the destination matrix. The performance characteristics of the 3D FFT implementation are further analyzed in Sec. 6.2.

4.3 Overlap Exchanges

The gradient or derivative calculation on a subdomain can be performed only after gathering the most recent data from all its neighbors. This is accommodated by exchanging the overlap regions between neighboring subdomains before every such operation, see the orange bars in Fig. 2. The size of these transfers range from N_d^3 to $N_x \times N_y \times N_d$ grid points, where N_d is the overlap size and $N_{\{x,y,z\}}$ the subdomain edge length.

Since the overlaps have to contain the most recent data, it is difficult to properly hide the communication by overlapping it with useful computation. However, our implementation structurally allows to hide up to 50% of the communication by exploiting stages where multiple arrays have to be updated at the same time. This is achieved by a combination of persistent communications and non-blocking calls provided by MPI, see Listing 1.

```
1  // Initialization stage
2  for (auto &m: /* Velocity matrices U_x, U_y, U_z */) {
3    for (auto &n: m.getNeighbors()) {
4      MPI_Send_init(n.data, n.size, n.otherRank, /* ... */);
5      MPI_Recv_init(n.data, n.size, n.otherRank, /* ... */);
6    }
7  }
8
9  // Main simulation loop stage
10 for (auto &m: /* Velocity matrices U_x, U_y, U_z */)
11   MPI_Startall(m.getRequests().size(), m.getRequests().data());
12
13 for (auto &m: /* Velocity matrices U_x, U_y, U_z */) {
14   // Partially overlapped communication
15   MPI_Waitall(m.getRequests().size(), m.getRequests().data(),
16               /* ... */);
17   Compute_Forward_FFT_3D(m);
18 }
```

Listing 1: The principle of the communication overlap during the velocity gradient calculation. A set of persistent communications are created during the initialization stage (Line 1–7). The overlap exchange for multiple matrices is started in the simulation loop (Line 9–11). As soon as the communication for a given matrix has finished, the computation starts. The other transfers can be still in flight (Line 13–17).

4.4 Parallel Input and Output

The simulation of nonlinear ultrasound wave propagation in large realistic domains naturally requires a fast and scalable input-output subsystem. Not only may typical medium, source and input signal descriptions occupy tens to hundreds of GBs, the simulation outputs in the form of pressure and velocity time series can easily spread over a few TBs [16].

The parallel I/O subsystem of the k-Wave toolbox is based on the Parallel HDF5 library [6] supported by the MPI-IO back-end [4]. In combination with the parallel LustreFS file system, very good throughput reaching up to 15 GB/s has been achieved on several clusters [10].

Although the parallel I/O is not intended to be deeply investigated in this paper, there is a key observation that needs to be mentioned because of it being a significant source of issues on the accelerated part of the Salomon supercomputer. The obstacle is the lack of LustreFS support in the Xeon Phi implementation of the Intel MPI library. This makes the use of the distributed scratch file system impossible and forces us to use an NFS mounted home file system, not primarily intended for parallel I/O. As a consequence, the amount of data being collected was severely limited to avoid any inference in the experiments.

5 Scaling Results

5.1 Overview

The main objective of this section is to investigate the performance and scaling properties of the cluster of Intel Xeon Phi accelerators in simulating ultrasound wave propagation. The secondary objective is the evaluation of the software support and the ease of use of the whole platform.

The experiments were conduced on various numbers of Salomon's accelerated nodes ranging from 1 up to 256 (512 accelerators). Since Salomon's hypercube interconnection topology has some blocking factor, there is a measurable variation between the job instances stemming from the different placement of the MPI processes on currently available nodes. Although the job placement can be restricted manually, it significantly prolongs the waiting time in the queues and thus was not used. Instead, particular benchmark runs of the same type were packed into a single large job to maintain fair conditions. Different job allocations were used for benchmarking subdomains placed over CPUs and accelerators. Therefore, a small variation between these benchmarks may be observed, however, it is considered insignificant from the perspective of the overall scaling trends and even the absolute performance.

Every benchmark run consisted of 100 time steps of the simulation loop summarized in Fig. 2. This number is deemed sufficient to hide any cache and communication warm-up effects. All experiments, if not stated otherwise, were performed with the most typical overlap size of 16 grid points.

The following subsections first focus on obtaining the performance and scaling results for the largest possible simulation domains while trying to work around the stability and HW/SW support issues. However, some issues are related to high numbers of nodes used for large simulation domain sizes. It was not possible to resolve these issues even after consultation with the Salomon support and the vendor. In the second subsection, we thus limited the simulation domain size and the number of nodes used to mimic an ideal situation. Finally, the most significant stages of the KSTD solver are analyzed and discussed.

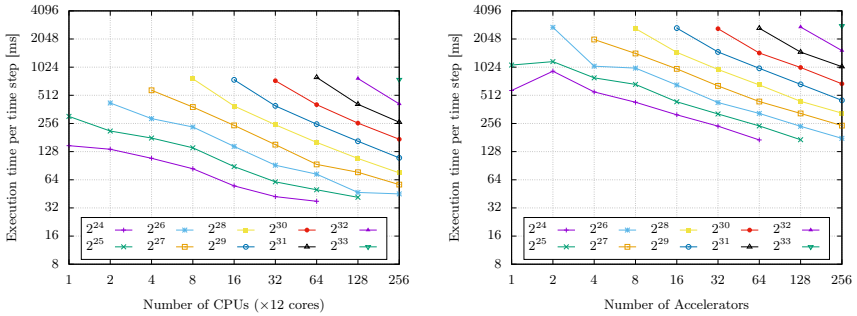
5.2 Performance Scaling on Large Domains

Driven by the practical demands from industry and medical physics, we first focus on the performance scaling of large simulation domains spread over many accelerators. The simulation domain sizes of interest are expanded from $256 \times 256 \times 256$ (2^{24}) to $2048 \times 2048 \times 2048$ (2^{33}) grid points by sequentially doubling the dimension sizes starting from the least significant one. The domains are partitioned into a number of subdomains growing from 1 to 256. The numbers of subdomains for particular domain sizes are further restricted by the size of the smallest meaningful subdomain (64^3) and the largest possible subdomain ($256 \times 256 \times 512$) that can fit within memory, excluding the overlaps. Particular subdomains are assigned either to a single accelerator or a single CPU socket. This allows us to mutually compare performance scaling of both architectures.

Figures 4 and 5 show the strong and weak scaling achieved by both architectures. Although the whole range of overlap sizes between 2 to 32 grid points was investigated, only the most common overlap size of 16 is presented for the sake of brevity. The scaling of small overlap sizes generally does better due to a higher degree of communication overlapping. For bigger overlap sizes, the absolute execution time is more influenced by the communication time elaborated in Fig. 8 and the strong scaling curves appear flatter.

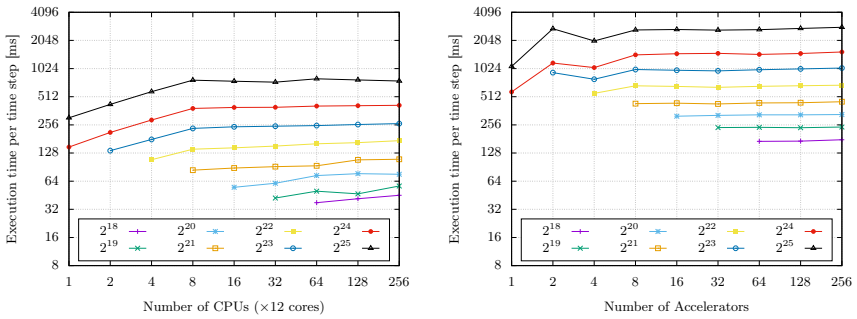
A brief glance at Fig. 4 reveals a significant disparity between the performance of CPUs and accelerators. The dramatic slowdown on the accelerator side is caused by the communication layer, more specifically the Intel MPI Infiniband backend DAPL [15], which becomes unstable for more than 32 accelerators in a single job. This problem was discussed in detail with the cluster support team and the vendor specialists but has not been resolved yet. The only solution that proved to be stable was to replace the 56 Gbit Infiniband by a 1 Gbit Ethernet interface. This typically leads to a 4.3× slower execution compared to CPUs. Nevertheless the code maintains reasonable strong scaling factors of 1.45 (average speedup achieved by doubling the computational resources) comparable with the execution on CPUs. The only reason why the performance slump is not even deeper is the limited performance of the FFTs on the accelerators, and therefore, better opportunity for communication overlapping.

Examining further the CPU and accelerator scaling plots, there are very few anomalies in the scaling curves. The most significant are apparent for very small numbers of subdomains when using accelerators, or for tiny subdomains when using CPUs. Putting these results into context of the global Fourier basis decompositions (GFB) presented in [13], the LFB implementation on CPUs shows its superiority with typical and peak speedup of 2 and 6, respectively.



(a) CPUs with FDR Infiniband (b) Accelerators with 1 Gbit Ethernet

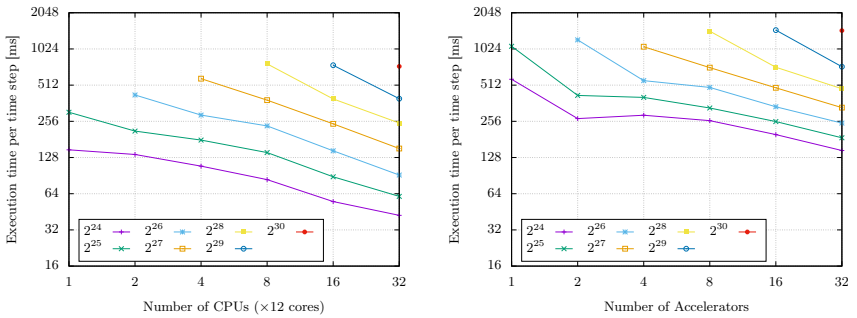
Fig. 4: Strong scaling evaluation on large domains having between 2^{24} and 2^{33} grid points with an overlap size of 16 grid points collected on CPUs and accelerators. Since the Infiniband interface is not stable for more than 32 accelerators, 1 Gbit Ethernet is used instead.



(a) CPUs with FDR Infiniband (b) Accelerators with 1 Gbit Ethernet

Fig. 5: Weak scaling evaluation on large subdomains with overlap size of 16 grid points collected on CPUs and accelerators. The size of the subdomains ranges between 2^{18} and 2^{25} grid points.

Figure 5 shows the weak scaling achieved on the CPUs and accelerators. Each of the plotted series corresponds to a constant subdomain size from the investigated range between 64^3 and $256 \times 256 \times 512$ grid points. At first glance, poor weak scaling is observed when the simulation domain is split into less than 8 subdomains. This is due to the growing rank of the domain decomposition and the number of neighbors. Once a full 3D decomposition is reached, the scaling curves remain almost flat in-line with almost perfect scaling.

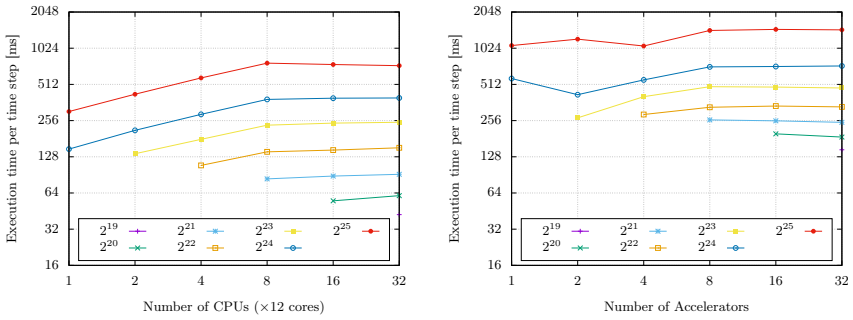


(a) CPUs with FDR Infiniband (b) Accelerators with FDR Infiniband

Fig. 6: Strong scaling evaluation on small domains (2^{24} to 2^{30} grid points) with an overlap size of 16 grid points collected on CPUs and accelerators, both supported by FDR Infiniband.

5.3 Performance Scaling on Small Domains

In order to better quantify the impact of the misbehaving Infiniband on large jobs, the previous experiments were repeated with reduced domain sizes (up to 1024^3) and a reduced number of accelerators (up to 32). Figure 6 shows the improved strong scaling on accelerators when using Infiniband and compares it again with the CPU baseline. Not only does Infiniband reduce the overall execution time by a factor of 1.9 with respect to Ethernet, it also improves the scaling factors from 1.45 to 1.52. The reduction of the communication overhead has also a positive impact on weak scaling presented in Fig. 7. Here, the penalty caused by increasing the rank of the decomposition is greatly reduced and very good scaling is achieved even when going from one to two subdomains. Nevertheless, the final conclusion is that a cluster of Intel Xeon Phi accelerators is significantly slower than a comparable cluster of CPUs, even though the theoretical parameters of the architecture promise the direct opposite.

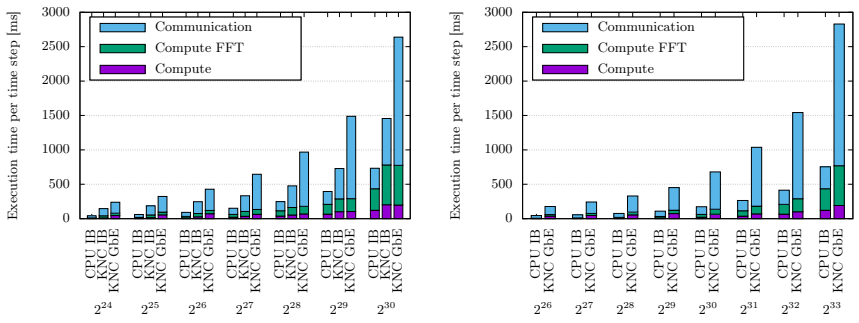


(a) CPUs with FDR Infiniband (b) Accelerators with FDR Infiniband

Fig. 7: Weak scaling evaluation on small domains (subdomains of 2^{19} to 2^{25} grid points) with an overlap size of 16 grid points collected on CPUs and accelerators, both supported by FDR Infiniband.

5.4 Simulation Time Breakdown

Figure 8 shows the execution time breakdown for domains of various sizes partitioned into 32 ($2 \times 4 \times 4$) and 256 ($4 \times 8 \times 8$) subdomains with an overlaps size of 16 grid points. The slower interconnection of the accelerators becomes immediately evident from Fig. 8b which shows that this usually comprises more than 60% of the compute time. By comparing the communication time on the accelerators with CPUs we can find a massive 3× to 12× deterioration which increases proportionally with the simulation size. On the other hand, the calculation time remains favorable for medium sized subdomains. For small ones, there is not enough work for all 120 threads used. On contrary the L2 cache is



(a) 32 CPUs or Accelerators (b) 256 CPUs or Accelerators

Fig. 8: The execution time breakdown for a time step of the simulation loop collected on 32 and 256 CPUs (CPU) or accelerators (KNC) for different domain sizes with an overlap size of 16 grid points. Results for both the Infiniband (IB) and the 1 Gbit Ethernet (GbE) interconnects are shown.

exceeded for subdomains bigger than 256^3 grid points, which leads to a significant drop in the FFT performance (see 8a at 2^{29} and 2^{30}). In conclusion, the computation can only be 1.8× slower than a single CPU socket for favorable subdomain sizes while it can worsen to more than 16× slower for large subdomains. Both interconnect and FFT computation issues are further investigated in Sec. 6.

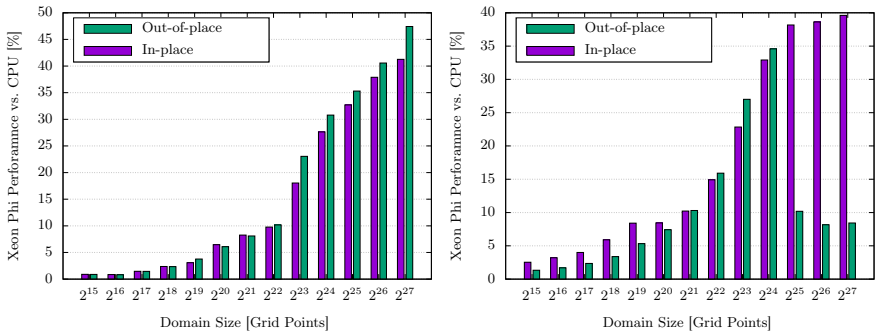
6 Platform Investigation

6.1 Overview

As mentioned earlier, the practical use of a cluster of Intel Xeon Phi accelerators suffers from many issues and immature libraries. Apart from the missing support for the LustreFS file system and the instability of the Intel MPI communication back-end for the infiniband interface, there are two other major issues. The first is the performance of 3D FFTs and the second one is the communication bandwidth. Both are further investigated in this section.

6.2 Performance of 3D FFTs on Intel Xeon Phi

Since the performance breakdown presented in Fig. 8 revealed relatively poor performance of the FFTs running on the accelerators compared to a single CPU, a deeper performance investigation was carried out. The particular routines of interest were the real-to-complex (R2C) and complex-to-real (C2R) 3D FFTs with the conjugate-even storage within an interleaved complex array implemented in the Intel MKL library. Both in-place and out-of-place transforms were evaluated.



(a) 3D real-to-complex transform (b) 3D complex-to-real transform

Fig. 9: Performance of a single accelerator vs. a single CPU while running in-place and out-of place forward and inverse 3D FFTs implemented by Intel MKL.

Figure 9 shows the relative performance of a single accelerator executing 120 threads versus a single CPU executing 12 threads. The thread counts were chosen according to the best performance achieved by each architecture. The measured performance is plotted for several domain sizes growing from 32^3 to 512^3 grid points.

Although the theoretical values suggest an accelerator should be four times faster than a single CPU, our FFT benchmarks revealed a different picture. For small domains, the performance of the accelerators is dismal, reaching less than 10% compared to a single CPU. This is very likely caused by poor workload distribution among cores and cache coherency issues such as false sharing. However, even a 3D transform over a 64^3 domain requires the execution of 3×4096 independent 1D FFTs, which appears to be enough to employ 120 threads evenly. Moreover, as the size of each 1D FFT is only 64 elements (256 B, 1 MB in total), the capacity of L2 cannot be a true bottleneck. That being said, the 1D transforms are unlikely to cause the troubles. The other sources of performance issues are the multi-dimensional matrix transpositions and thread synchronization between particular phases of the 3D FFT algorithm.

With an increasing transform size, the performance of the accelerator increases, reaching 50% of the CPU performance in the best case. Furthermore, the performance of the forward transforms is slightly better than the performance of the inverse one. Much more unexpected behavior is observed for the out-of-place transforms on domains larger than 256^3 grid points. Beyond this size, the relative performance of the accelerator falls below 10%. At this size, the data cannot fit within L2 cache any more. Considering the only difference between the forward and inverse FFT being the sign in the exponent, this behavior must be a hidden bug inside the complex-to-real out-of-place FFTs in MKL. This statement is supported by the measurements provided by Intel VTune performance counters. At a size of $256 \times 256 \times 512$ grid points, there is a 60-fold increase in the L1 data

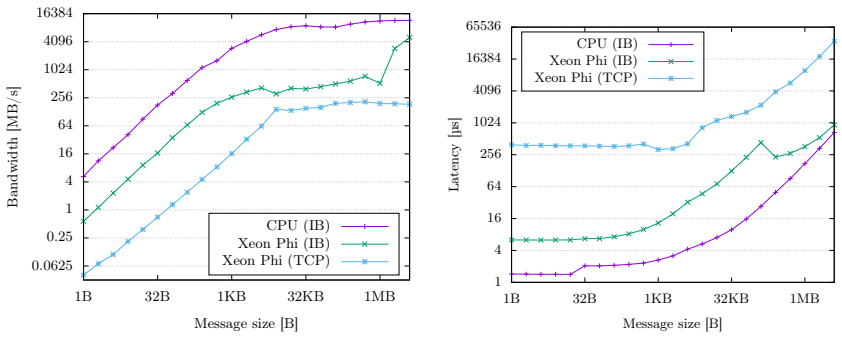
cache misses, most of which end up as L2 data cache memory fills. These misses seem to be caused by the Read for Ownership (RFO) operations executed before memory writes recorded as `L2_DATA_WRITE_MISS_MEM_FILL` events, a typical sign of false sharing. The issue is related to the FFT execution plan and the memory hierarchy as out-of-place real-to-complex transforms do not exhibit similar behavior.

6.3 Performance of Intel MPI on Intel Xeon Phi

The second issue afflicting the performance is the interconnection. Even when using the Infiniband interface, the accelerators do not achieve comparable communication times to CPUs as shown in Fig. 8a.

Figure 10 shows the bidirectional bandwidth and latency in CPU-to-CPU and Phi-to-Phi communication over the Infiniband and Ethernet interconnects measured by the OSU Micro-Benchmark suite[4]. The difference is obvious. Not only is there an order of magnitude lower bandwidth for small messages when the infiniband is used between accelerators, the loss is not caught up even for medium sized messages. There appears to be an improvement for 4 MB messages, however, this is helpful only for the biggest subdomains with an overlap size of 32 grid points. The communication latency copies the same trend, being typically 5× to 10× longer. The explanation of this behavior can be found in the combination of the PCI-Express 2.0 (about 16 GB/s bidirectional), additional hop between PCI-Express 2.0 and PCI-Express 3.0 where the Infiniband card is connected to, and low attainable memory bandwidth of a single Intel Xeon Phi core of (only about 2.65 GB/s) yielding 4.1 GB/s for bidirectional communication.

The bandwidth of the Ethernet interface is, as expected, limited by the network interface for big messages.



(a) Inter-node Bidirectional bandwidth. (b) Inter-node Latency.

Fig. 10: Comparison of the communication bandwidth and latency for processors with Infiniband, and accelerators with Infiniband and TCP over 1 Gbit Ethernet.

[4] `http://mvapich.cse.ohio-state.edu/benchmarks/`

7 Conclusion

The goal of this paper was to investigate the performance scaling and suitability of accelerated clusters based on Intel Xeon Phis for large simulations of ultrasound wave propagation in biologically relevant materials, and compare these results with a common CPU cluster.

In order to keep low requirements on the spatial and temporal resolution, the k-space corrected pseudospectral method from the k-Wave toolbox was used [27]. The communication overhead is reduced by the local Fourier basis decomposition [14] and the communication is overlapped with the computation in more than 50% of cases.

The performance was measured on a set of domain sizes starting from tiny ones and growing to the edge of practical feasibility. During testing, many problems were encountered. The most significant one is the Infiniband network instability caused by the DAPL back-end in the Intel MPI. This bug made the use of the native Infiniband impossible for jobs spreading over more than 32 accelerators. Despite careful investigation and collaboration with the Salomon support, this issue has not been resolved yet. The only solution to get large simulations to work was to use a service Ethernet network. This naturally has a large impact on performance. Regardless, the scaling was similar to the underlying cluster of CPUs, which is caused by another issue related to the FFT performance, and the absolute execution time is typically 4.3× longer. This is far from the expected performance inferred from the theoretical parameters. When the size of the simulation is limited to fit within 32 accelerators, the Infiniband interconnection remains stable. This yields much better performance, but still almost 1.9× lower than CPUs.

The second big issue is the performance of 3D FFTs, which in some configurations reaches only a fraction of the CPU performance. This is very likely caused by a bug in the MKL library related to false sharing during the matrix transposition between particular 1D FFTs. The best performance obtained reached about 50% of the CPU performance.

Despite all the mentioned troubles, we are still positive about the code deployment on the accelerated cluster. The main motivation is the accounting policy on Salomon, where the use of accelerators is now for free. This allows us to run large batches of independent simulations. In the future, we would also like to implement a load-balancing algorithm that would allow us to use both the CPUs and the accelerators in a single simulation.

8 Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602" and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing

Center - LM2015070". This project has received funding from the European Union's Horizon 2020 research and innovation programme H2020 ICT 2016-2017 under grant agreement No 732411 and is an initiative of the Photonics Public Private Partnership. This work was further supported in part by the Engineering and Physical Sciences Research Council (EPRSC), UK, grant numbers EP/L020262/1 and EP/S026371/1.

References

1. Beard, P.: Biomedical photoacoustic imaging. Interface Focus **1**(4), 602–631 (aug 2011)
2. Boyd, J.P.: A Comparison of Numerical Algorithms for Fourier Extension of the First, Second, and Third Kinds. Journal of Computational Physics **178**(1), 118–160 (may 2002)
3. Boyd, J.P.: Asymptotic Fourier Coefficients for a C∞ Bell (Smoothed-"Top-Hat") & the Fourier Extension Problem. Journal of Scientific Computing **29**(1), 1–24 (oct 2006)
4. Coloma, K., Ching, A., Choudhary, A., Liao, W.k., Ross, R., Thakur, R., Ward, L.: A New Flexible MPI Collective I/O Implementation. In: 2006 IEEE International Conference on Cluster Computing. pp. 1–10. IEEE (2006)
5. Dubinsky, T.J., Cuevas, C., Dighe, M.K., Kolokythas, O., Joo, H.H.: High-intensity focused ultrasound: Current potential and oncologic applications. American Journal of Roentgenology **190**(1), 191–199 (jan 2008)
6. Folk, M., Heber, G., Koziol, Q., Pourmal, E., Robinson, D.: An overview of the HDF5 technology suite and its applications. In: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases - AD '11 (2011)
7. Frigo, M., Johnson, S.G.: The Design and Implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005)
8. Gholami, A., Hill, J., Malhotra, D., Biros, G.: AccFFT: A library for distributed-memory FFT on CPU and GPU architectures (May 2016)
9. Gu, J., Jing, Y.: Modeling of wave propagation for medical ultrasound: a review. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control **62**(11), 1979–1992 (nov 2015)
10. Howison, M., Koziol, Q., Knaak, D., Mainzer, J., Shalf, J.: Tuning HDF5 for Lustre file systems. IASDS '10 Proceedings of the Workshop on Interfaces and Abstractions for Scientific Data Storage **5** (2012)
11. Intel Corporation: Math Kernel Library 11.3 Developer Reference. Intel Corporation (2015)
12. Israeli, M., Vozovoi, L., Averbuch, A.: Spectral multidomain technique with local Fourier basis. J. Sci. Comput. **8**(2), 135–149 (1993)
13. Jaros, J., Rendell, A.P., Treeby, B.E.: Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound. The International Journal of High Performance Computing Applications **30**(2), 137–155 (2016)
14. Jaros, J., Vaverka, F., Treeby, B.E.: Spectral domain decomposition using local fourier basis: Application to ultrasound simulation on a cluster of gpus. Supercomputing Frontiers and Innovations **3**(3), 40–55 (2016)
15. Jeffers, J., Reinders, J.: Intel Xeon Phi Coprocessor High Performance Programming. No. 1, Elsevier Inc., Waltham (2013)

16. Klepárník, P., Bařina, D., Zemčík, P., Jaroš, J.: Efficient low-resource compression of hifu data. Information **9**(7), 1–14 (2018). https://doi.org/10.3390/info9070155, https://www.fit.vut.cz/research/publication/11764
17. Mast, T., Souriau, L., Liu, D.L., Tabei, M., Nachman, A., Waag, R.: A k-space method for large-scale models of wave propagation in tissue. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control **48**(2), 341–354 (mar 2001)
18. Meairs, S., Alonso, A.: Ultrasound, microbubbles and the blood–brain barrier. Progress in Biophysics and Molecular Biology **93**(1-3), 354–362 (jan 2007)
19. Nandapalan, N., Jaros, J., Treeby, B.E., Rendell, A.P.: Implementation of 3d ffts across multiple gpus in shared memory environments. In: Proceedings of the Thirteenth International Conference on Parallel and Distributed Computing, Applications and Technologies. pp. 167–172 (2012)
20. Nikl, V., Jaros, J.: Parallelisation of the 3d fast fourier transform using the hybrid openmp/mpi decomposition. In: Mathematical and Engineering Methods in Computer Science. pp. 100–112. LNCS 8934, Springer International Publishing (2014)
21. Pekurovsky, D.: P3DFFT: A Framework for Parallel Computations of Fourier Transforms in Three Dimensions (2012)
22. Pinton, G.F., Dahl, J., Rosenzweig, S., Trahey, G.E.: A heterogeneous nonlinear attenuating full-wave model of ultrasound. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control **56**(3), 474–488 (2009)
23. Pippig, M.: PFFT-An extension of FFTW to massively parallel architectures. SIAM Journal on Scientific Computing **35**(3), 213–236 (2013)
24. Sorensen, H., Jones, D., Heideman, M., Burrus, C.: Real-valued fast Fourier transform algorithms. IEEE Transactions on Acoustics, Speech, and Signal Processing **35**(6), 849–863 (jun 1987)
25. Tabei, M., Mast, Douglas, T., Waag, R.C.: A k-space method for coupled first-order acoustic propagation equations. The Journal of the Acoustical Society of America **111**(1 Pt 1), 53–63 (jan 2002)
26. Tomov, S., Haidar, A., Ayala, A., Schultz, D., Dongarra, J.: Fft-ecp fast fourier transform (2019-01 2019)
27. Treeby, B.E., Jaros, J., Rendell, A.P., Cox, B.T.: Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method. The Journal of the Acoustical Society of America **131**(6), 4324–36 (2012)
28. Tufail, Y., Yoshihiro, A., Pati, S., Li, M.M., Tyler, W.J.: Ultrasonic neuromodulation by brain stimulation with transcranial ultrasound. Nature Protocols **6**(9), 1453–1470 (sep 2011)
29. Vaverka, F., Treeby, B.E., Jaros, J.: Evaluation of the suitability of intel xeon phi clusters for the simulation of ultrasound wave propagation using pseudospectral methods. In: Computational Science – ICCS 2019, vol. 11538, pp. 577–590. Springer International Publishing
30. Wang, E., Zhang, Q., Shen, B., Zhang, G., Lu, X., Wu, Q., Wang, Y.: High-Performance Computing on the Intel Xeon Phi. Springer International Publishing (2014)
31. Yu, W., Mittra, R., Su, T., Liu, Y., Yang, X.: Parallel Finite-Difference Time-Domain Method. ARTECH HOUSE, INC., Norwood, MA 02062 (2006)