Evaluation of the Suitability of Intel Xeon Phi Clusters for the Simulation of Ultrasound Wave Propagation Using Pseudospectral Methods

Filip Vaverka[1(✉)], Bradley E. Treeby[2], and Jiri Jaros[1]

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

[2] Medical Physics and Biomedical Engineering, Biomedical Ultrasound Group,
University College London, Malet Place Eng Bldg, London WC1E 6BT, UK
b.treeby@ucl.ac.uk

Abstract. The ability to perform large-scale ultrasound simulations using Fourier pseudospectral methods has generated significant interest in medical ultrasonics, including for treatment planning in therapeutic ultrasound and image reconstruction in photoacoustic tomography. However, the routine execution of such simulations is computationally very challenging. Nowadays, the trend in parallel computing is towards the use of accelerated clusters where computationally intensive parts are offloaded from processors to accelerators. During last five years, Intel has released two generations of Xeon Phi accelerators. The goal of this paper is to investigate the performance on both architectures with respect to current processors, and evaluate the suitability of accelerated clusters for the distributed simulation of ultrasound propagation using Fourier-based methods. The paper reveals that the former version of Xeon Phis, the Knight's Corner architecture, suffers from several flaws that reduce the performance far below the Haswell processors. On the other hand, the second generation called Knight's Landing shows very promising performance comparable with current processors.

Keywords: Ultrasound simulations · Pseudospectral methods · k-Wave toolbox · Intel Xeon Phi · KNC · KNL · MPI · OpenMP · Performance evaluation · Scaling

1 Introduction

There are many medical applications of ultrasound ranging from ultrasound and photoacoustic imaging [2] through to neurostimulation and neuromodulation [23] to direct treatment using high intensity focused ultrasound (HIFU) [5,14]. The common characteristic of all these applications is the reliance on fast, accurate

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11538, pp. 577–590, 2019.
https://doi.org/10.1007/978-3-030-22744-9_45

and versatile ultrasound propagation models in biological tissue [17]. A typical scenario consists of modeling a nonlinear ultrasound wave propagating from one or more ultrasound sources through a heterogeneous medium with a power law absorption and eventually recorded by one or more ultrasound sensors or within a given region of interest.

Computational speed is still a concern even though supercomputing facilities are used. The fundamental issue is the size of the computational domain compared to the highest wavelength modeled. This challenge has raised a lot of interest across the ultrasound, mathematics and high performance computing communities. As a consequence, several ultrasound modeling packages have been released, see [8] for a recent review.

One promising approach to discretizing the acoustic governing equations is the pseudospectral time-domain (PSTD) and k-space pseudospectral time-domain (KSTD) methods [21]. The main benefit is the exponential convergence with increasing spatial resolution which can significantly reduce memory requirements for large 3D simulations. The KSTD method is considered more accurate than the PSTD method because it uses a semi-analytical time-stepping schemes [20], whereas the pseudospectral method uses a finite-difference approximation. Consequently, the KSTD method allows for a larger time step.

Unfortunately, the relaxation in the required discretization for the PSTD and KSTD schemes compared to conventional finite-difference schemes is somewhat counteracted by the introduction of a global trigonometric basis and the use of the fast Fourier transform (FFT) to compute spatial gradients. For PSTD schemes, the FFTs are one-dimensional (in the direction of the required gradient). However, for the KSTD scheme, the introduction of the k-space correction means the FFTs are performed in three-dimensions. The scaling on parallel systems is then inherently limited by the necessity of performing distributed matrix transpositions over all subdomains [11] as part of the 3D FFT. Although a lot of work on efficient distributed FFTs has been carried out (FFTW [6], P3DFFT [16], PFFT [18], AccFFT [7] or multi-GPU CUDA FFT [15]), 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.

A promising direction in joining the advantages of FDTD and PSTD methods is the decomposition of the global Fourier basis into a set of local ones [10]. This composition inherits the simplicity of the FDTD nearest neighbor halo exchange while maintaining the spectral accuracy of PSTD and KSTD methods [22].

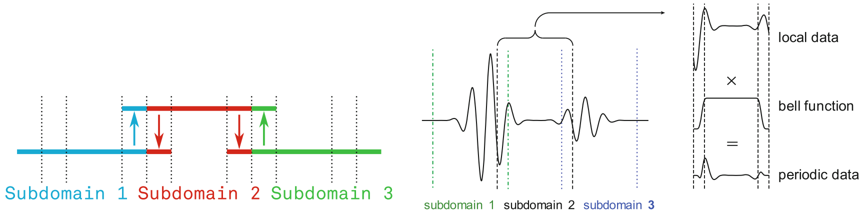
This paper investigates the suitability of domain decomposition, implemented as part of the k-Wave toolbox [12], for deployment on cluster of Intel Xeon Phi accelerators based on both Knight's Corner (KNC) and Knight's Landing (KNL) architectures. First, the principle of local Fourier basis domain decomposition vital for the distributed computation is explained in Sect. 2. Second, the architecture of two accelerated clusters is described in Sect. 3. After that, the main components of the benchmark implementation are outlined in Sect. 4. Section 5 presents the experimental results collected on both clusters and compares the performance scaling with a CPU cluster. Finally, the most important conclusions are drawn.

2 Efficient Local Fourier Basis Domain Decomposition

The local Fourier basis domain decomposition (LFB for short) of PSTD and KSTD methods splits the 3D domain into a number of cuboid subdomains, each of which is supported by its own local Fourier basis [10]. The required global communication is consequently reduced into local direct neighbor exchange of the overlap regions. However, the split of the Fourier basis breaks the periodicity condition on local domains. To restore it, Fourier extension methods can be used [3]. The subdomains are coupled by overlap exchanges and the local subdomain periodicity is restored by multiplying with a bell function [12], see Fig. 1.

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 [4]. While the overlap size can be chosen by the user as a compromise between the accuracy and the performance for any particular problem, the shape of the bell function has be optimized in advance for the whole set of overlap sizes by means of numerical optimization.

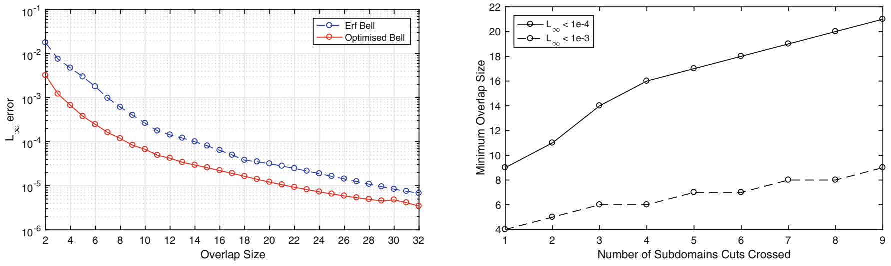
Figure 2 shows the relationship between the accuracy, the size of the overlap, and the bell function used. Figure 2a shows the dependency of the numerical error in terms of the L_∞ norm on the size of the overlap for the domain split into two subdomains (a single cut). The figure also compares two different bell functions, the well known Error (Erf) function [1] and a numerically optimized one. Figure 2b indicates the minimum overlap size required to keep the error below 10^{-3} or 10^{-4} for a given number of subdomain interfaces the wave has to cross in a single Cartesian direction. The conclusion drawn from this figure suggests that an overlap size of 8 or 20 should be chosen to keep the overall accuracy of 10^{-3} and 10^{-4} for decompositions with the total number of subdomains below 512 (8 in every Cartesian direction), respectively.



(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 an overlap from both neighboring subdomains. These overlaps are periodically exchanged. (b) After the exchange, each local subdomain is multiplied by a bell function.



(a) Single interface (b) Overlap size to meet a given error

Fig. 2. The accuracy of the local Fourier basis domain decomposition determined by the size of the overlap, the number of subdomain interfaces and the shape of the bell function.

The numerical model of the nonlinear wave propagation in heterogeneous absorption medium investigated in this paper is based on the governing equations derived by Treeby [21] written as three-coupled first-order partial differential equations:

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

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 isentropic sound speed, and B/A is the nonlinearity parameter. Two linear source terms (force $\mathbf{F}$ and mass $\mathbf{M}$) are also included.

The computation itself consists of an iterative algorithm running over a given number of time steps (a detailed description is given in [11,12]). Each time step is composed of a sequence of element-wise operations, overlap exchanges and

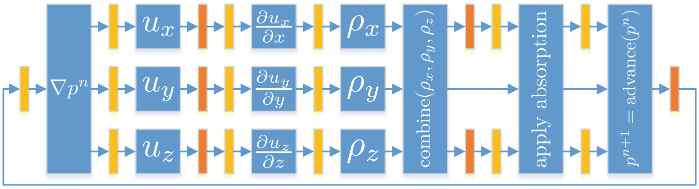

∇p^n
u_x
u_y
u_z
$\frac{\partial u_x}{\partial x}$
$\frac{\partial u_y}{\partial y}$
$\frac{\partial u_z}{\partial z}$
ρ_x
ρ_y
ρ_z
combine(ρ_x, ρ_y, ρ_z)
apply absorption
p^{n+1} = advance(p^n)

Fig. 3. Simplified computation loop governed by Eq. (1). The blue blocks denote element-wise operations, yellow 3D FFTs, and orange the overlap exchanges. (Color figure online)

local 3D FFTs, see Fig. 3 [12]. The majority of the computation time is usually spent on 3D FFTs or overlap exchanges.

3 Target Architecture

The target architectures of interest are represented by two clusters of Intel Xeon Phi accelerators. Salomon is an accelerated cluster based on the first generation of Knight's Corner architecture operated by the IT4Innovations national supercomputing center Ostrava, Czech Republic[1]. CoolMUC3 is a newer cluster based on the second generation of the Knight's Landing architecture operated by the Leibniz Rechenzentrum in Garching, Germany[2].

3.1 Architecture of Knight's Corners Cluster

Salomon consists of 1008 compute nodes, 432 of which are accelerated by Intel Xeon Phi 7120P accelerators. The architecture of Salomon's accelerated part is shown in Fig. 4. 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 processor sockets and accelerators 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 the 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).

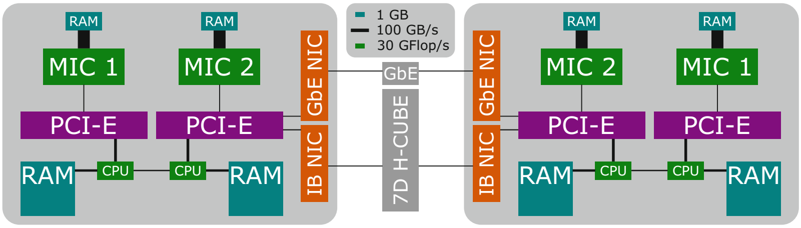


Fig. 4. The architecture of the Salomon accelerated nodes and interconnection. The size of the rectangles representing individual components is proportional to their performance, bandwidth or capacity.

[1] https://docs.it4i.cz/salomon/hardware-overview/.
[2] https://www.lrz.de/services/compute/linux-cluster/coolmuc3/overview_en/.

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 processing unit (VPU). The KNC cores are supported by 30.5 MB of L2 cache 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 [13]. 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 is theoretically supposed to 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 Salomon reaches one PFLOP/s.

3.2 Architecture of Knight's Landing Cluster

CoolMUC3 consists of 148 nodes equipped with Intel Xeon Phi 7210-F accelerators. The architecture of CoolMUC3 is shown in Fig. 5. The Xeon Phi generation installed in this system is the first from Intel to be stand-alone. Since a classic CPU is thus not required to control the computation node, the cluster is composed of single socket nodes populated with the KNL processors only.

The nodes are interconnected by an Intel Omnipath network forming a fat tree topology. A single node has two independent NICs connected via PCI-Express 3.0 x16 interfaces offering aggregated throughput of 25 GB/s per node.

Every KNL chip consists of 32 tiles placed in a 2D grid providing a bisection bandwidth of 700 GB/s. Each tile integrates two out-of-order 4-wide SMT cores, four 512-bit wide vector processing units and 1 MB of shared L2 cache. The theoretical performance of a single KNL chip exceeds 5 TFLOP/s in single precision. The main memory of each node is split between 16 GB of High Bandwidth Memory (HBM) and 96 GB of DDR4 providing bandwidth of 460 GB/s and 80 GB/s, respectively. Since the KSTD and PSTD solvers are proven to be memory bound, only HBM memory is used in this study. The expected speedup

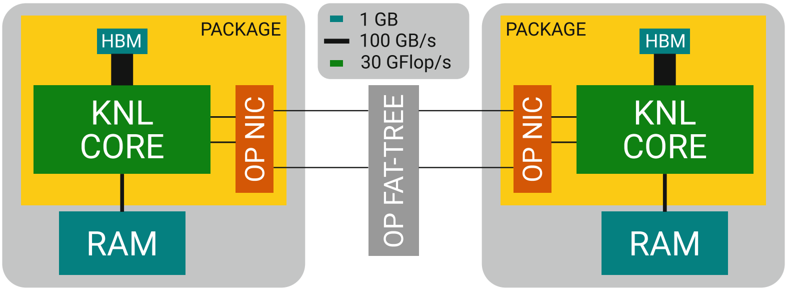


Fig. 5. The architecture of the CoolMUC3 accelerated nodes and interconnection. The size of the rectangles representing individual components is proportional to their performance, bandwidth or capacity.

over a Haswell CPU should attain a factor of 10 for compute bound and 6 for memory bound applications.

4 Implementation

The PSTD and KSTD methods (e.g., as implemented in k-Wave) are typical examples of memory bound problems with a relatively low arithmetic intensity, usually on the order of $O(\log n)$ (due to FFTs). Furthermore, the LFB domain decomposition relies on communication stages which are latency sensitive because very little communication can be overlapped. Such a combination of algorithm properties suggests the use of parallel architectures with high memory bandwidth and, ideally, a direct access to NICs. The Intel Xeon Phi accelerators look very favorable from this point of view.

The proposed implementation can be executed on both CPUs and accelerators. Although, it is possible to use any combinations of CPUs and Xeon Phis concurrently, this is not tested in this paper because no load balancing has been implemented yet (only uniform decompositions are supported). The code is logically structured into MPI processes handling single subdomains running on particular accelerators or CPUs. The work distribution within the subdomain is implemented by means of OpenMP threads and OpenMP SIMD constructs. Since realistic simulations do not require double precision, only single precision floating point operations are used in the critical path. This yields higher performance and saves valuable memory bandwidth.

Logically, the simulation code of k-Wave code boils down to a mix of element-wise operations on 3D real or complex matrices, 3D Fourier transforms and overlap exchanges. These are further explained in following subsections.

4.1 Element-Wise Operations

The element-wise operations can easily take the full advantage of the accelerator memory bandwidth and compute power because of their locality. Listing 1 shows a typical example of an element-wise computation kernel.

```
const float norm = 1.0f / (Nx * Ny * Nz);

#pragma omp parallel for collapse(2)
for(size_t z = 0; z < Nz; z++) {
  for(size_t y = 0; y < Ny; y++) {
    const float ePmlY = pmlY[y];

    #pragma omp simd
    for(size_t x = 0; x < Nx; x++) {
      const size_t i = z * Ny * Nx + y * Nx + x;
      uy[i] = ((uy[i] * ePmlY) - (norm * fftPressure[i])) * ePmlY;
    }
  }
}
```

Listing 1: Update of the acoustic velocity field in the y-axis direction with an application of the perfectly matched layer (PML) optimized for Intel Xeon Phi.

The kernel runs over three spatial dimensions starting from the most significant one to comply with the row-major array ordering. The two outermost loops are collapsed into a single one and parallelised over multiple OpenMP threads. Although the loop collapsing introduces some overhead into the calculation of z and y indices, it is vital for even distribution of the work among many parallel threads. Let us note that up to 256 threads can be executed simultaneously on the Xeon Phi while the maximum subdomain size which can fit within the accelerator memory is on the order of 400^3 grid points. The innermost loop is vectorised by means of an OpenMP SIMD pragma to ensure the full utilization of the vector units.

4.2 Fourier Transforms

The most computationally expensive part of the simulation loop consists of 14 3D fast Fourier transforms calculated over the local subdomains. Their actual implementation relies on third party libraries compatible with the FFTW interface [6], in this case the Intel MKL[3] library [9] which is believed to be well optimized for the Intel Xeon Phi architecture [24].

The algorithms to perform forward and inverse FFTs typically assume complex input and output data. However, the solutions of the wave equation require only real-valued data in the time domain. This makes the use of real-to-complex (R2C) and complex-to-real (C2R) transforms possible and reduces the temporal and spatial complexity of the FFT by a factor of two [19].

The simulation code naturally uses out-of-place transforms to preserve the input fields needed later in the simulation loop and calculates derivatives in reusable temporary matrices. Unfortunately, the implementation of the out-of-place C2R transforms in the MKL library has proved to be very inefficient on KNC, showing an almost 12× performance drop for bigger subdomains. Hence, the C2R transforms are performed in-place using a temporary matrix and the results consequently copied to the destination matrix.

4.3 Overlap Exchanges

Before every gradient calculation, it is necessary to synchronize all subdomains by exchanging the overlap regions, see the orange bars in Fig. 3. Depending on the rank of the decomposition, up to 26 mutual exchanges are performed per subdomain. The amount of data being transferred is proportional to the size of the overlap region, and also dependent on the mutual position of subdomains in the simulated space. If two subdomains only touch at the corner, only a small number of grid points is transferred (${N_d}^3$, where d is the size of the overlap). In contrast, if two subdomains sit side by side, a large block of $N_x \times N_y \times N_d$ grid points must be transferred.

Since the overlaps have to contain the most recent data, it is difficult to hide the communication by overlapping it with useful computation. Fortunately, it is

[3] Intel 2017b and 2018a suite were used on Salomon and CoolMUC3 respectively.

possible to decouple the calculation of velocity gradients for each spatial dimension and overlap the data exchange with gradient calculations. This enables two out of three communications to be hidden. The same approach is applied to the medium absorption calculation where two gradients are calculated independently. In total, up to 50% of the communication time may be hidden.

Practically speaking, the communication overlapping is achieved by a combination of persistent communications and non-blocking calls provided by MPI. Listing 2 shows the principle of the communication hiding during the velocity gradient calculation. The calculation of partial derivative of the velocity along a given axis starts as soon as the overlaps arrive while the other communication can still be in flight.

```
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 2: The principle of communication hiding during velocity gradient calculation. Persistent communications are created in the initialization stage (lines 1–7). The exchange on multiple matrices is started at a given place in the simulation loop (lines 9–11). As soon as the communication for a given matrix finishes, the computation starts. The other transfers can be still in flight (lines 13–17).

5 Experimental Results

Numerical experiments were conduced on a various number of accelerators ranging from 1 to 16. The number of accelerators was limited by our preparatory allocations enabling the maximum use of 16 nodes on CoolMUC3 and the capacity of the express queue on Salomon (8 nodes, 2 accelerators each).

Benchmark runs of the same type were packed into single larger jobs to maintain the same MPI rank placement over the cluster between particular benchmark runs. Therefore, only a tiny variation, considered insignificant from the perspective of the overall scaling trends and even the absolute performance, may be observed between different benchmark runs. Every benchmark run consists of 100 time steps of the simulation loop summarized in Fig. 3. This number is deemed sufficient to hide any cache and communication warm-up effects.

The simulation domain was progressively expanded from 256×256×256 (2^{24}) to 1024×1024×512 (2^{29}) grid points by sequentially doubling the dimension sizes starting from the least significant one. The global domains were partitioned into a number of subdomains growing from 1 to 16. The numbers of subdomains for particular domain sizes were 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 the memory, excluding the overlaps. Particular subdomains were assigned either to a single accelerator or a single CPU sockets. The reason for this kind of comparison is twofold. First, the amount of communication overhead is kept the same and IT4I's allocation, and second, pricing policies take only CPU cores into account (and not the accelerator usage).

On the OpenMP level, each subdomain was processed by the optimal number of threads on a given architecture. For Haswell CPUs we used one thread per core (12 threads per CPU) whereas the optimal number of threads for a single KNC accelerator was found to be 120 (2 threads per core). Finally, KNL performed best using all 256 threads per accelerator.

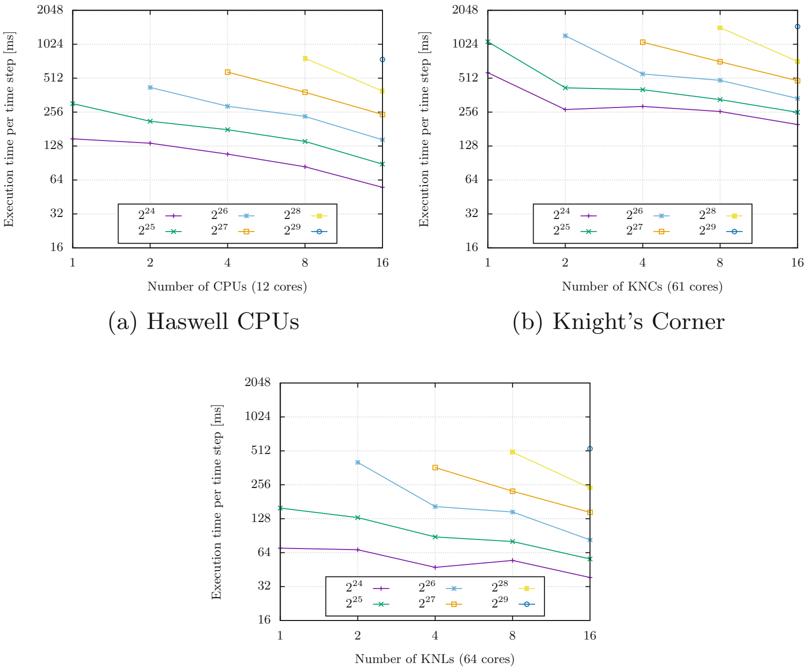
5.1 Strong Scaling Evaluation

Figure 6 shows the strong scaling for investigated architectures. Although the whole range of the overlap sizes between 2 to 32 grid points was investigated, only one overlap size of 16 is presented for the sake of brevity. Scaling with small overlap sizes generally runs faster due to a higher degree of communication overlapping. For bigger overlap sizes, the absolute execution time is more influenced by the communication time and the strong scaling curves appear flatter.

Looking at Figs. 6a and b, a significant disproportion in the performance between CPUs and KNC accelerators can be observed. The execution time on KNC is between 2.2× and 4.3× longer than on CPUs. This behavior was further investigated by analyzing flat performance profiles. First, the overlap exchange among accelerators is on average 2× slower than among CPUs. This substantial overhead can be attributed to a combined effect of the additional PCI-Express communication and much slower compute cores on the accelerators responsible for packing the overlaps into MPI messages and their management. The maximum measured core-to-core bandwidth only reaches 2.65 GB/s, which is about a half of the theoretical Infiniband bandwidth. Second, the performance of the 3D FFTs very low. For the subdomain sizes examined in this section, the speedup of KNC with respect to CPU was between 0.03 for domain sizes of 64^3 and 0.4 for domain sizes of 256^3. For small domains, this is most certainly due to expected thread congestion and cache coherence effects such as false sharing. Since the Intel MKL is a closed software, it was impossible to further investigate this issue.

Apart from the poor absolute performance of KNC, the scaling factors look favorable. For the three biggest domains, the scaling factor reaches a value of 1.52 every time the number of accelerators is doubled. This yields a parallel efficiency of 76%, which is comparable to the CPU cluster.

The strong scaling achieved by the KNL cluster is significantly better, see Fig. 6c. The KNL accelerators are significantly faster than the previous generation KNC. When only a single accelerator is used, the benchmarks are completed in an order of magnitude shorter time. When communication comes into play,



(a) Haswell CPUs

(b) Knight's Corner

(c) Knight's Landing

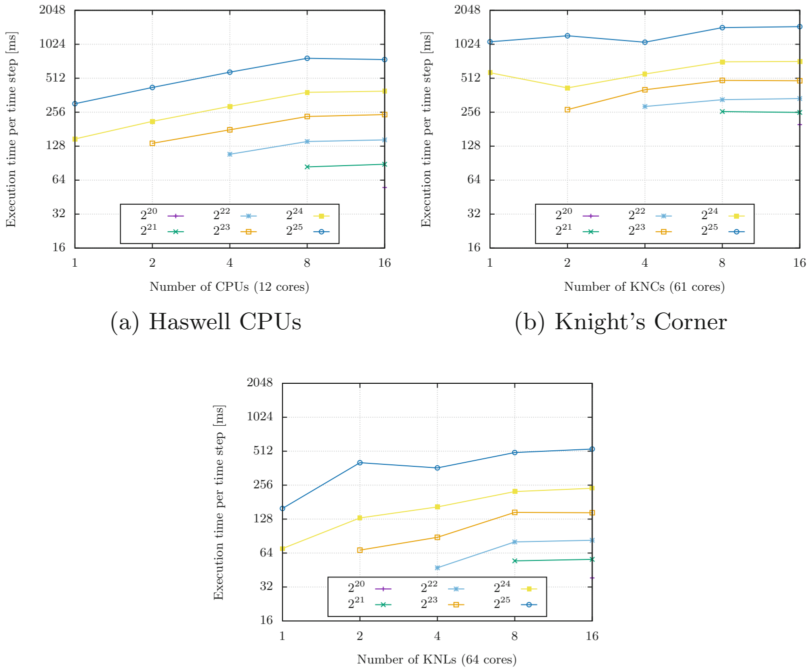
Fig. 6. Strong scaling with overlap size of 16 grid points.

the Omnipath interconnection shows its strengths. The average scaling factors reaches 1.62. This amounts to 4.16× speedup compared to KNCs with the Infiniband interconnect on Salomon. The comparison against the Haswell CPUs with 12 cores yields an average speedup of 1.7 in favor of the new Intel Xeon Phi accelerators.

5.2 Weak Scaling Evaluation

Figure 7 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 $128 \times 128 \times 64$ and $256 \times 256 \times 512$ grid points. At first glance, poor weak scaling is observed for CPUs and KNLs when the simulation domain is split into fewer than 8 subdomains. This is due to the growing rank of the domain decomposition and the number of neighbors. Since the computation on KNC is much slower, there is better possibility for communication overlapping and the initial growth in the execution time is not observed.

Once a full 3D decomposition is reached, the scaling curves remain almost flat being a sight of almost perfect scaling. However, to support this statement, benchmark results using a much higher number of accelerators are needed.



(a) Haswell CPUs

(b) Knight's Corner

(c) Knight's Landing

Fig. 7. Weak scaling with overlap size of 16 grid points.

6 Conclusion

The goal of this paper was to investigate the performance scaling and suitability of two Xeon Phi accelerated clusters for large simulations of ultrasound wave propagation using Fourier pseudospectral methods and compare the computational performance against a common CPU cluster.

Starting with the former Knight's Corner architecture of Intel Xeon Phi, we conclude that the cluster of KNCs did not come up to expectations when running the pseudospectral time domain solver of the k-Wave toolbox [12]. The biggest obstacle was the performance of the 3D fast Fourier transforms, which for the domain sizes of interest reaches only a fraction of the performance provided by CPU. This may be caused by a too many active threads when small domains are computed, and relatively small L2 caches resulting in many accesses to the main memory if the domain sizes are bigger. In future work, we would like to examine other FFT libraries such as FFTW [6] and confirm that the poor performance is caused by a bug in the Intel MKL library. Considering that this issue might be fixed in the future, the strong and weak scaling achieved on KNC promises easy deployment on all of the 432 Salomon's accelerators. Since the allocation policy in terms of core hours charged is very favorable for accelerators, the cluster of KNC can decrease the computational costs for running large scale simulations.

The performance of the Knight's landing cluster was (after the experience with its predecessor) a pleasant surprise. The performance of a single KNL accelerator is 4× higher than KNC and almost 1.7× higher than a single twelve core Haswell CPU. The strong scaling is also better with a parallel efficiency of 81%. This shows that Intel has achieved a significant improvement on the interconnection part as well. In the future work, we would like to use much higher number of the KNL accelerators to extend the scaling study and run full production simulations on CoolMUC3.

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 also supported by the Engineering and Physical Sciences Research Council, UK, grant numbers EP/L020262/1 and EP/P008860/1.

References

1. Andrews, L.C.: Special Functions of Mathematics for Engineers. SPIE Pub. (1997)
2. Beard, P.: Biomedical photoacoustic imaging. Interface Focus **1**(4), 602–631 (2011). https://doi.org/10.1098/rsfs.2011.0028
3. Boyd, J.P.: A comparison of numerical algorithms for fourier extension of the first, second, and third kinds. J. Comput. Phys. **178**(1), 118–160 (2002). https://doi.org/10.1006/jcph.2002.7023
4. Boyd, J.P.: Asymptotic fourier coefficients for a C∞ bell (Smoothed-"Top-Hat") & the fourier extension problem. J. Sci. Comput. **29**(1), 1–24 (2006). https://doi.org/10.1007/s10915-005-9010-7
5. Dubinsky, T.J., Cuevas, C., Dighe, M.K., Kolokythas, O., Joo, H.H.: High-intensity focused ultrasound: current potential and oncologic applications. Am. J. Roentgenol. **190**(1), 191–199 (2008)
6. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proc. IEEE **93**(2), 216–231 (2005)
7. Gholami, A., Hill, J., Malhotra, D., Biros, G.: AccFFT: a library for distributed-memory FFT on CPU and GPU architectures, May 2016. http://arxiv.org/abs/1506.07933v3
8. Gu, J., Jing, Y.: Modeling of wave propagation for medical ultrasound: a review. IEEE Trans. Ultrason. Ferroelectr. Freq. Control **62**(11), 1979–1992 (2015). https://doi.org/10.1109/TUFFC.2015.007034
9. Intel Corporation: Math Kernel Library 11.3 Developer Reference. Intel Corporation (2015)
10. Israeli, M., Vozovoi, L., Averbuch, A.: Spectral multidomain technique with local Fourier basis. J. Sci. Comput. **8**(2), 135–149 (1993)
11. Jaros, J., Rendell, A.P., Treeby, B.E.: Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound. J. High Perform. Comput. Appl. **30**(2), 137–155 (2016)

12. Jaros, J., Vaverka, F., Treeby, B.E.: Spectral domain decomposition using local fourier basis: application to ultrasound simulation on a cluster of GPUs. Supercomput. Frontiers Innov. **3**(3), 40–55 (2016)
13. Jeffers, J., Reinders, J.: Intel Xeon Phi Coprocessor High Performance Programming, vol. 1. Elsevier Inc., Waltham (2013)
14. Meairs, S., Alonso, A.: Ultrasound, microbubbles and the blood-brain barrier. Progress Biophys. Mol. Biol. **93**(1–3), 354–362 (2007)
15. Nandapalan, N., Jaros, J., Treeby, E.B., AlistairRendell, 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). https://doi.org/10.1109/PDCAT.2012.79. http://www.fit.vutbr.cz/research/view_pub.php?id=10171
16. Pekurovsky, D.: P3DFFT: a framework for parallel computations of Fourier transforms in three dimensions (2012). https://doi.org/10.1137/11082748X
17. Pinton, G.F., Dahl, J., Rosenzweig, S., Trahey, G.E.: A heterogeneous nonlinear attenuating full-wave model of ultrasound. IEEE Trans. Ultrason. Ferroelectr. Freq. Control **56**(3), 474–488 (2009)
18. Pippig, M.: PFFT: an extension of FFTW to massively parallel architectures. SIAM J. Sci. Comput. **35**(3), 213–236 (2013)
19. Sorensen, H., Jones, D., Heideman, M., Burrus, C.: Real-valued fast Fourier transform algorithms. IEEE Trans. Acoust. Speech Signal Process. **35**(6), 849–863 (1987). https://doi.org/10.1109/TASSP.1987.1165220
20. Tabei, M., Mast, T.D., Waag, R.C.: A k-space method for coupled first-order acoustic propagation equations. J. Acoust. Soc. Am. **111**(1 Pt 1), 53–63 (2002). https://doi.org/10.1121/1.1421344
21. 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. J. Acoust. Soc. Am. **131**(6), 4324–4336 (2012). https://doi.org/10.1121/1.4712021
22. Treeby, B.E., Vaverka, F., Jaros, J.: Performance and accuracy analysis of nonlinear k-wave simulations using local domain decomposition with an 8-GPU server. Proc. Meet. Acoust. **34**(1), 022002 (2018)
23. Tufail, Y., Yoshihiro, A., Pati, S., Li, M.M., Tyler, W.J.: Ultrasonic neuromodulation by brain stimulation with transcranial ultrasound. Nat. Protoc. **6**(9), 1453–1470 (2011). https://doi.org/10.1038/nprot.2011.371
24. Wang, E., et al.: High-Performance Computing on the Intel® Xeon Phi™. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06486-4