

Seismic Wave Propagation Simulations on Low-power and Performance-centric Manycores

Márcio Castro, Emilio Francesquini, Fabrice Dupros, Hideo Aochi, Philippe Navaux, Jean-François Mehaut

► To cite this version:

Márcio Castro, Emilio Francesquini, Fabrice Dupros, Hideo Aochi, Philippe Navaux, et al.. Seismic Wave Propagation Simulations on Low-power and Performance-centric Manycores. Parallel Computing, Elsevier, 2016, 10.1016/j.parco.2016.01.011 . hal-01273153

HAL Id: hal-01273153 https://hal.archives-ouvertes.fr/hal-01273153

Submitted on 12 Feb 2016 $\,$

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Seismic Wave Propagation Simulations on Low-power and Performance-centric Manycores

Márcio Castro^a, Emilio Francesquini^b, Fabrice Dupros^c, Hideo Aochi^c, Philippe O. A. Navaux^d, Jean-François Méhaut^e

^aDepartment of Informatics and Statistics, Federal University of Santa Catarina (UFSC) Campus Reitor João David Ferreira Lima, Trindade, 88040-970, Florianópolis, Brazil ^bInstitute of Computing, University of Campinas (UNICAMP)

Av. Albert Einstein, 1251, Cidade Universitária, 13083-852, Campinas, Brazil [°]BRGM

BP 6009, 45060 Orléans Cedex 2, France

^dInformatics Institute, Federal University of Rio Grande do Sul (UFRGS)
 Av. Bento Gonçalves, 9500, Campus do Vale, 91501-970, Porto Alegre, Brazil
 ^eCEA-DRT - LIG Laboratory, University of Grenoble Alpes
 110 Avenue de la Chimie, 38400 Saint-Martin d'Hères, France

Abstract

The large processing requirements of seismic wave propagation simulations make High Performance Computing (HPC) architectures a natural choice for their execution. However, to keep both the current pace of performance improvements and the power consumption under a strict power budget, HPC systems must be more energy efficient than ever. As a response to this need, energyefficient and low-power processors began to make their way into the market. In this paper we employ a novel low-power processor, the MPPA-256 manycore, to perform seismic wave propagation simulations. It has 256 cores connected by a NoC, no cache-coherence and only a limited amount of on-chip memory. We describe how its particular architectural characteristics influenced our solution for an energy-efficient implementation. As a counterpoint to the low-power MPPA-256 architecture, we employ Xeon Phi, a performance-centric manycore. Although both processors share some architectural similarities, the challenges to implement an efficient seismic wave propagation kernel on these platforms are very different. In this work we compare the performance and energy efficiency of our implementations for these processors to proven and optimized solutions for other hardware platforms such as general-purpose processors and a GPU. Our experimental results show that MPPA-256 has the best energy efficiency, consuming at least 77% less energy than the other evaluated platforms, whereas the performance of our solution for the Xeon Phi is on par with a state-of-the-art solution for GPUs.

Keywords: Seismic wave propagation; MPPA-256; Xeon Phi; HPC; performance; energy efficiency

Preprint submitted to Parallel Computing

Email addresses: marcio.castro@ufsc.br (Márcio Castro),

 $[\]verb|francesquini@ic.unicamp.br| (Emilio Francesquini), \verb|f.dupros@brgm.fr| (Fabrice Dupros), \\$

1 1. Introduction

Public policies for risk mitigation and damage assessment in hypothetical earthquakes scenarios as well as oil and gas exploration make an intensive use of simulations of large-scale seismic wave propagation. In order to provide realistic results, these simulations rely on complex models, which often demand considerable computational resources to reliably process vast amounts of data r in a reasonable amount of time.

⁸ Such processing capabilities are often provided by High Performance Com-⁹ puting (HPC) platforms. Up until recently, HPC platform capabilities were ¹⁰ evaluated almost exclusively based on their raw processing speed. However, one ¹¹ of the aspects that hinder the quest for ever growing performance is the exces-¹² sive use of energy. For that reason, the energy efficiency of HPC platforms has ¹³ become, in some contexts, as important as their raw performance.

Indeed, the seek for alternatives to lower current energy consumption arose 14 first in the embedded systems community but lately became one of the ma-15 jor concerns of the HPC scientific community [1, 2]. Recently, manycore pro-16 cessors, a new class of highly-parallel chips, was unveiled. Tilera Tile-Gx [3], 17 Kalray MPPA-256 [4], Adapteva Epiphany-IV [5], Intel Single-Chip Cloud Com-18 puter (SCC) [6] and Xeon Phi are examples of such processors, providing up to 19 hundreds of autonomous cores that can be used to accomplish both data and 20 task parallelism. This distinctive characteristic sets them apart from SIMD-like 21 highly-parallel architectures such as Graphics Processing Units (GPUs). For 22 this reason, in this paper, we classify GPUs separately, in their own category. 23

While some manycore processors may present better energy efficiency than 24 state-of-the-art general-purpose multicore processors [7], their particular archi-25 tectural characteristics make the development of efficient scientific parallel ap-26 plications a challenging task [5, 8]. Some of these processors are built and opti-27 mized for certain classes of embedded applications like signal processing, video 28 decoding and routing. Additionally, processors such as MPPA-256 have impor-29 tant memory constraints, e.q., limited amount of directly addressable memory 30 and absence of cache coherence protocols. Furthermore, communication costs 31 between cores are not uniform, they depend on the location of the commu-32 nicating cores. One of the main reasons for this difference in communication 33 cost is the Network-on-Chip (NoC). When applications distribute work among 34 the cores of the processor, they should take into consideration the placement 35 of these tasks and their communication patterns to minimize communication 36 costs. This problem is similar to that faced by applications running on NUMA 37 platforms [9, 10], or MPI applications that try to take into account both the 38 network topology and the memory hierarchy to improve performance [11, 12]. 39 In this paper we outline the architectural distinctiveness of the low-power 40

⁴¹ MPPA-256 manycore processor. Considering these characteristics, we describe ⁴² how the main kernel of a seismic wave propagation simulator was adapted to

h.aochi@brgm.fr (Hideo Aochi), navaux@inf.ufrgs.br (Philippe O. A. Navaux), jean-francois.mehaut@imag.fr (Jean-François Méhaut)

this platform. Due to the limited size of the local memories in MPPA-256, we
developed a new multi-level tiling strategy and a prefetching mechanism to
allow us to deal with real simulation scenarios and to alleviate communication
overheads. We also describe the difficulties and solutions (some of them generic
enough to be used in different contexts) we employed during this adaptation.

As a counterpoint to the low-power MPPA-256, we also adapted the same 48 kernel to a performance-centric manycore, the Xeon Phi. Although both pro-49 cessors share some architectural characteristics such as the presence of a NoC, 50 non-uniform costs of communication between cores and limited fast local mem-51 ories, they have their own peculiarities which directed us to employ different 52 approaches in each case. For instance, in both architectures, each thread has 53 access to approximately 128 kB of fast local memory. However, their memory 54 organization is very different. On Xeon Phi each core has 512 kB of L2 shared 55 by up to 4 threads. On the other hand, on MPPA-256 the processing cores have 56 8 kB of L1 cache each and cores are grouped into clusters of 16. Each one of 57 these clusters has 2 MB, of work memory shared among the cores. Contrary to 58 Xeon Phi, MPPA-256 does not have cache-coherence or automatic load/store of 59 values from/to main memory, and all memory transferences must be handled 60 by the programmer explicitly. Ultimately, this means that in order to be able 61 to achieve good performance on MPPA-256, the application is obliged to tightly 62 control communications between the cores in the same cluster, between clusters, 63 and between each cluster and the main memory. This paper explains the de-64 tails that made us decide for a multi-level tiling strategy proposed for MPPA-256 65 instead of a more traditional approach of cache blocking that was used for Xeon 66 67 Phi

Taking as a basis optimized GPU and general-purpose processor implemen-68 tations for this kernel, we show that even if MPPA-256 presents an increased 69 software development complexity, it can indeed be used as an energy-efficient 70 alternative to perform seismic wave propagation simulations. Our results show 71 that the solution we propose for the MPPA-256 processor achieves the best en-72 ergy efficiency, consuming 77%, 86% and 88% less energy than optimized so-73 lutions for GPU, general-purpose and Xeon Phi architectures, respectively. The 74 performance achieved on Xeon Phi, on the other hand, is comparable to a state-75 of-the-art GPU. Moreover, the execution was 58% and 73% faster than that of 76 multicores and MPPA, respectively. 77

The remainder of this paper is organized as follows. Section 2 discusses the fundamentals of seismic wave propagation. Then, Section 3 discusses the main challenges we must overcome when dealing with parallel seismic wave propagations on MPPA-256 and Xeon Phi. Moreover, it presents our approaches to perform efficient seismic wave propagation simulations on both processors. Section 4 discusses performance and energy efficiency experimental results. Section 5 describes related work and Section 6 concludes this paper.

85 2. Seismic Wave Propagation

⁸⁶ Quantitative earthquake hazard assessment is crucial for public policy, re-⁸⁷ duction of future damages and urban planning. Recent important geological

events, such as the $M_W 7.9$ (moment magnitude) earthquake in China (2008) or 88 the $M_W 8.8$ earthquake in Chile (2010), have been studied using numerical tools 89 with good agreement with observation. From a physical point a view, the mech-90 anisms used to describe earthquake events are complex and highly nonlinear. 91 For the sake of simplicity, wave propagation simulations usually consider elastic 92 domains and include some anelasticity to account for attenuations. However, 93 in the case of strong motion, these simplifying assumptions are not capable of 94 correctly describing the phenomenon. A fundamental challenge for earthquake 95 engineering is predicting the level and variability of strong ground motion from 96 future earthquakes. Enhancing this predictive ability requires understanding of 97 the earthquake source, the effects of the propagation path on the seismic waves, 98 and basin and near-surface site effects. 99

In this paper, we restrict our discussion to the elastodynamics equations 100 corresponding to the elastic situation using a linear model. However, our ap-101 proach could also be used in nonlinear models that rely on a linear solution 102 inside the iterative procedure such as Newton-like methods. Among numeri-103 cal methods, variational methods such as finite element method [14] or spectral 104 element method [15] can be employed for both regional and global scale seismol-105 ogy whereas the Discontinous Galerkin method is very efficient for earthquake 106 source modeling [16]. In terms of popularity, the Finite Difference Method 107 (FDM) remains one of the top contenders due to its implementation simplicity 108 and efficiency for a wide range of applications. Although this method was ini-109 tially proposed almost thirty years ago [17], this classic discretization technique 110 111 has been the focus of continuous refinements since its proposal [18, 19].

112 Simulations of seismic wave propagations are often constrained by the computational and storage capacity of the hardware platform. Thus seismologists 113 often reduce the scope of the simulation to a small volume instead of considering 114 the whole Earth. The memory footprint of each simulation depends on both 115 the domain of interest and the number of grid points per wavelength. This last 116 condition guarantees numerical stability and therefore the accuracy of the sim-117 ulation. In our case, for practical reasons, we limit our simulations to problem 118 domains that fit into the 2 GB of memory available on the MPPA-256 platform. 119 For that, we perform regional scale modeling spanning a few hundred kilometers 120 in each spatial direction. 121

In the next section we present the standard sequential algorithm for seismic wave propagation simulations. *Aochi et al.* [20] provide a more detailed description of the equations governing these simulations in the case of an elastic material.

¹²⁶ 2.1. Discretization and Standard Sequential Algorithm

As mentioned before, the FDM is one of the most popular techniques to solve the elastodynamics equations and to simulate the propagation of seismic waves. One of the key features of this scheme is the introduction of a staggeredgrid [17] for the discretization of the seismic wave equation, which in the case of an elastic material is:

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial j} + F_i \tag{1}$$

Additionally, the constitutive relation in the case of an isotropic medium is:

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda \delta_{ij} \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) + \mu \left(\frac{\partial v_i}{\partial j} + \frac{\partial v_j}{\partial i} \right) \tag{2}$$

¹³³ Where indices i, j, k represent a component of a vector or tensor field in ¹³⁴ cartesian coordinates $(x, y, z), v_i$ and σ_{ij} represent the velocity and stress field ¹³⁵ respectively, and F_i denotes an external source force. ρ is the material density ¹³⁶ and λ and μ are the elastic coefficients known as Lamé parameters. A time ¹³⁷ derivative is denoted by $\frac{\partial}{\partial t}$ and a spatial derivative with respect to the *i*-th ¹³⁸ direction is represented by $\frac{\partial}{\partial i}$. The Kronecker symbol δ_{ij} is equal to 1 if i = j¹³⁹ and zero otherwise.

Indeed, all the unknowns are evaluated at the same location for classical
collocated methods over a regular Cartesian grid whereas the staggered grid
leads to a shift of the derivatives by half a grid cell. The equations are rewritten
as a first-order system in time and therefore the velocity and the stress fields
can be simultaneously evaluated at a given time step.

Exponents *i*, *j*, *k* indicate the spatial direction with $\sigma^{ijk} = \sigma(i\Delta s, j\Delta s, k\Delta s)$, Δs corresponds to the space step and Δt to the time step. The elastic coefficients ρ , μ et λ are defined at location (i, j, k). Starting at time step $t = \ell \Delta t$, the following unknowns corresponding to the next time step $t = (\ell + \frac{1}{2})\Delta t$ are computed:

$$v_x^{(i+\frac{1}{2})jk} \left(l + \frac{1}{2} \right), \quad v_y^{i(j+\frac{1}{2})k} \left(l + \frac{1}{2} \right), \quad v_z^{ij(k+\frac{1}{2})} \left(l + \frac{1}{2} \right)$$
(3)

Then, at time $t = (\ell + 1)\Delta t$, the following unknowns are computed:

$$\sigma_{xx}^{ijk}(l+1), \quad \sigma_{yy}^{ijk}(l+1), \quad \sigma_{zz}^{ijk}(l+1), \quad \sigma_{xy}^{(i+\frac{1}{2})(j+\frac{1}{2})k}(l+1), \\ \sigma_{zy}^{i(j+\frac{1}{2})(k+\frac{1}{2})}(l+1), \quad \sigma_{zx}^{(i+\frac{1}{2})j(k+\frac{1}{2})}(l+1)$$
(4)

For instance, the stencil applied for the computation of the velocity component in the x-direction is given by:

$$\begin{aligned} v_x^{(i+\frac{1}{2})jk} \left(l + \frac{1}{2} \right) &= v_x^{(i+\frac{1}{2})jk} \left(l - \frac{1}{2} \right) + a_1 F_x^{(i+\frac{1}{2})jk} \\ &+ a_2 \Big[\frac{\sigma_{xx}^{(i+1)jk} - \sigma_{xx}^{ijk}}{\Delta x} + \frac{\sigma_{xy}^{(i+\frac{1}{2})(j+\frac{1}{2})k} - \sigma_{xy}^{(i+\frac{1}{2})(j-\frac{1}{2})k}}{\Delta y} + \frac{\sigma_{xz}^{(i+\frac{1}{2})j(k+\frac{1}{2})} - \sigma_{xz}^{(i+\frac{1}{2})j(k-\frac{1}{2})}}{\Delta z} \Big] \\ &- a_3 \Big[\frac{\sigma_{xx}^{(i+2)jk} - \sigma_{xx}^{(i-1)jk}}{\Delta x} + \frac{\sigma_{xy}^{(i+\frac{1}{2})(j+\frac{3}{2})k} - \sigma_{xy}^{(i+\frac{1}{2})(j-\frac{3}{2})k}}{\Delta y} + \frac{\sigma_{xz}^{(i+\frac{1}{2})j(k+\frac{3}{2})} - \sigma_{xz}^{(i+\frac{1}{2})j(k-\frac{3}{2})}}{\Delta z} \Big] \end{aligned}$$

$$(5)$$

The computational procedure representing this evaluation is described in Algorithm 2.1. Inside the time step loop, the first triple nested loop is devoted to the computation of the velocity components, and the second loop reuses the velocity results of the previous time step to update the stress field.

 $\begin{array}{l} \textbf{Algorithm 2.1: SEQUENTIAL SEISMIC WAVE PROPAGATION}(\sigma, v) \\ \hline \textbf{for } x \leftarrow 1 \text{ to } x_dimension \\ \textbf{do} \begin{cases} \textbf{for } y \leftarrow 1 \text{ to } y_dimension \\ \textbf{do} & \{ \textbf{for } z \leftarrow 1 \text{ to } z_dimension \\ \textbf{do} & \{ \textbf{COMPUTE_VELOCITY}(\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}) \\ \textbf{for } x \leftarrow 1 \text{ to } x_dimension \\ \textbf{do} & \{ \textbf{for } z \leftarrow 1 \text{ to } z_dimension \\ \textbf{do} & \{ \textbf{for } z \leftarrow 1 \text{ to } z_dimension \\ \textbf{do} & \{ \textbf{for } z \leftarrow 1 \text{ to } z_dimension \\ \textbf{do} & \{ \textbf{COMPUTE_STRESS}(v_x, v_y, v_z) \\ \end{array} \right) \end{cases}$

The best parallelization strategy for the elastodynamics equations strongly depends on the characteristics of the underlying hardware architecture. In the following section, we detail the parallelization and optimization strategies we employed for the adaptation of this sequential algorithm to highly parallel manycore processors.

¹⁶² 3. Elastodynamics Numerical Kernel on Manycores

In this section we present our approaches to perform seismic wave propagation simulations on MPPA-256 and Xeon Phi. We first discuss in Section 3.1 some of the intrinsic characteristics and challenges that led us to employ different parallelization strategies and optimizations in each one of these processors. Then, in Sections 3.2 and 3.3, we describe in details our solutions for these hardware architectures.

169 3.1. Platforms and Challenges

MPPA-256. The MPPA-256 is a single-chip manycore processor developed 170 by Kalray that integrates 256 user cores and 32 system cores in 28 nm CMOS 171 technology running at 400 MHz. These cores are distributed across 16 compute 172 clusters and 4 I/O subsystems that communicate through data and control 173 NoCs. Each compute cluster has 16 cores called Processing Elements (PEs), 174 which are dedicated to run user threads (one thread per PE) in non-interruptible 175 and non-preemptible mode, and a low-latency shared memory of 2 MB. This 176 local memory enables a high bandwidth and throughput between PEs within the 177 same compute cluster. Each PE has private 2-way associative 32 kB instruction 178 and data caches. The system has no cache coherence protocol between PEs, 179 even among those in the same compute cluster. The board used in our tests 180



Figure 1: Manycore architectures.

has one of the I/O subsystems connected to an external LP-DDR3 of 2 GB.
Figure 1(a) shows an architectural overview of the MPPA-256 processor. Some
uses of this processor include embedded parallel signal processing and video
decoding [21], and parallel scientific applications [7].

The development of a seismic wave propagation kernel for this processor 185 can be a challenging task due to some of its intrinsic characteristics. First, the 186 low-latency memory available in each compute cluster acts as a cache, whose 187 goal is to store data retrieved from the DDR. However, data transfers between 188 the DDR and compute clusters' low-latency memories must be explicitly man-189 aged by the programmer, in other words, there are no automatic fetching or 190 prefetching mechanisms. Second, the amount of data needed to simulate real 191 wave propagation scenarios does not fit into the local memories. In fact, about 192 500 kB of memory is always in use by the operating system leaving about 1.5 MB193 free to house the application's data and code. For this reason, the application 194 has to take into account the data transfers and how these data should be sliced 195 in order to fit into the limited local memory. 196

Xeon Phi. Xeon Phi was designed to operate in a way similar to regular 197 x86 multicore processors. In this article we used Xeon Phi model 3120, which 198 features 57 cores. Each one of its cores has a clock frequency of 1.10 GHz, 199 32 kB for L1 instruction and data caches and 512 kB for L2 cache. Coherence 200 between the caches is guaranteed by hardware through a Global-distributed 201 Tag Directory (TD). Additionally, every core can directly address the shared 202 DDR memory (6 GB in our case) and is connected to the remaining cores by a 203 high-performance bidirectional ring-shaped NoC as shown in Figure 1(b). More-204 over, each core is 4-way multithreaded, *i.e.*, it is able to execute instructions 205 from four threads/processes (using time-multiplexed multithreading), helping to 206 reduce the effect of vector pipeline latency and memory access latencies. Differ-207 ently from other regular x86 multicore processors, each core has 512-bit vector 208 instructions, which makes this processor able to perform 16 single precision 209 operations, or 8 double precision operations, within a single instruction. 210

Porting a seismic finite difference numerical stencil to the Xeon Phi architecture requires a careful exploitation of two main aspects. The first one is related to the use of vector processing units. Since most of the performance power of

Xeon Phi comes from these units, it is essential to fully benefit from them. This 214 can be achieved by performing a clever decomposition of the 3D input problem 215 to maximize the use of long vectors in the unit-stride direction. The second 216 aspect is related to the L2 caches. When a core C_{src} accesses its L2 cache and 217 misses, an address request is sent to the tag directories throughout the ring. 218 If the requested data block is found in the cache of another core (C_{dst}) , it is 219 forwarded back through the ring to the L2 cache of C_{src} . Thus, the overhead 220 imposed by this protocol must be avoided whenever possible to improve appli-221 cation's performance. Overall, this can achieved by organizing data memory 222 accesses to improve data locality. 223

In the next sections we describe how the architectural distinctiveness of these architectures guided the development of a seismic wave propagation simulation kernel.

227 3.2. A Two-level Tiling Approach for the MPPA-256

The seismic wave propagation kernel has a high demand for memory band-228 width. This makes the efficient use of the low-latency memories distributed 229 among compute clusters indispensable. In contrast to standard x86 processors 230 in which it is not uncommon to find last-level cache sizes of tens of megabytes, 231 MPPA-256 has only 32 MB of low-latency memory divided into 2 MB chunks 232 spread throughout the 16 compute clusters. These chunks of memory are di-233 rectly exposed to the programmer that must explicitly control them. Indeed, 234 the efficiency of our algorithm relies on the ability to fully exploit these low-235 latency memories. To that end, we implemented a data fractioning strategy 236 that decomposes the problem into tiles small enough to fit into the memory 237 available on the compute clusters. Figure 2 illustrates the general idea of our 238 239 two-level tiling scheme.

The three-dimensional structures corresponding to the velocity and stress 240 fields are allocated on the DDR connected to the I/O subsystem to maximize 241 the overall problem size that can be simulated. Next, we divide the global 242 computational domain into several subdomains corresponding to the number 243 of compute clusters involved in the computation (Figure 2-1). This decom-244 position provides a first level of data-parallelism. To respect the width of the 245 stencil (fourth-order), we maintain an overlap of two grid points in each di-246 rection. These regions, called ghost zones, are updated at each stage of the 247 computation with point-to-point communications between neighboring clusters. 248 249 Unfortunately, only this first level of decomposition is not enough because the three-dimensional tiles do not fit into the 2 MB of low-latency memories 250 available to the compute clusters. A second level of decomposition is there-251 fore required. We performed this decomposition along the vertical direction as 252 we tile each three-dimensional subdomain into 2D slices (Figure 2-2). This 253 leads to a significant reduction of the memory consumption for each cluster but 254 requires maintaining a good balance between the computation and communica-255 tion. Our solution relies on a sliding window algorithm that traverses the 3D 256 domains using 2D planes and overlaps data transfers with computations. This 257 mechanism can be seen as an *explicit prefetching mechanism* (Figure 2-3) as 258



Figure 2: Two-level tiling scheme to exploit the memory hierarchy of MPPA-256.

259 2D planes required for the computation at one step are brought to the clusters during the computation performed at previous steps. To alleviate communication costs, asynchronous background data transfers are employed. Finally, OpenMP directives are employed inside each compute cluster to compute 2D problems with up to 16 PEs in parallel (Figure 2-4). Additional details about the implementation of the seismic wave propagation simulation kernel for the MPPA-256 architecture are provided by *Castro et al.* [8].

266 3.3. A Cache-aware Approach for the Xeon Phi

Research on the efficiency of seismic wave propagation simulations using the
FDM on multicore architectures has received a lot of attention lately [22, 23].
The performance of these simulations are typically memory-bound due to the
imbalance between the relatively fast point-wise computations and the intensive
memory accesses these computations require.

Several standard sequential implementations offer a poor cache reuse and 272 therefore achieve only a fraction of the peak performance of the processor. Even 273 if some optimizations alleviate this problem, the standard simulation algorithm 274 typically scans an array spanning several times the size of the cache using each 275 retrieved grid point only for a few operations. Therefore, the cost to bring the 276 needed data from the main memory to the fast local cache memories account 277 for an important share of the total simulation time, specially for a 3D problem. 278 In particular, this limitation is more noticeable in processors that feature 279 high core counts such as Xeon Phi due to the contention on the bidirectional 280 data ring. To improve the cache locality we employ a cache blocking technique. 281 Cache blocking is an optimization technique that intends to reduce memory 282 bandwidth bottlenecks. The main idea is to exploit the inherent data reuse 283 available in the triple nested loop of the elastodynamics kernel by ensuring that 284 data remains in cache across multiple uses. 285

Among classical blocking strategies, the approach described by Rivera and Tseng [24] proposes to tile two dimensions and to perform computations by accumulating the layers in the third one. This strategy is somehow similar to our tiling strategy proposed for the MPPA-256, since the idea is to control the data movement by prefetching the read and the write computing planes.

Although the sliding window algorithm and vectorial instructions are a good combination, their use on Xeon Phi is unfeasible in practice. The limited cache

size on Xeon Phi prevents the storage of three velocity and six stress components 293 in 512 kB of L2 cache memory shared by up to four threads. For that reason, 294 our implementation therefore relies on a 3D blocking strategy which leads to 295 a important speedup when we take the naïve implementation as a baseline as 296 shown in Section 4. This improvement is due to the reduction of the pressure on 207 the memory. We employ OpenMP to distribute the load throughout the different 298 computing cores and we use the OpenMP's loop collapse directive to improve 200 loop execution scheduling across the 3D domain decomposition. Additionally, 300 we consider low-level specific optimizations such as data alignment, vectorization 301 and thread affinity. On Xeon Phi, unaligned data severely limits the overall 302 performance. In our implementation, we align data with respect to the length 303 of the stencil (fourth order in our case) and we shift pointers in order fully 304 benefit from the Xeon Phi vectorization capabilities (16 single precision floats). 305 A similar optimization strategy can be found in [25]. Finally, we employ thread 306 affinity to ensure that threads are correctly bound to the cores to reduce memory 307 access latency and to alleviate the memory contention. 308

309 4. Experimental Results

In this section we evaluate the performance and energy consumption of our solutions to perform seismic wave propagation simulations on manycores. First, we briefly describe our evaluation methodology. Then, we analyze the impact of the optimizations on MPPA-256 and Xeon Phi. Finally, we carry out an overall comparison of the performance and energy consumption between our solutions for manycores and other standard optimized solutions for other modern platforms.

317 4.1. Methodology

We compare our solutions for the MPPA-256 and Xeon Phi to reference implementations of the same elastodynamics kernel for multicores [8, 26] and GPUs [8, 27]. These implementations were extracted from Ondes3D, a seismic wave propagation simulator developed by the French Geological Survey (BRGM). In this study we used the following platforms:

• MPPA-256: This low-power manycore processor integrates 16 compute 323 clusters in a single chip running at 400 MHz. Each compute cluster has 324 16 cores called PEs and a low-latency shared memory of 2MB shared 325 between all the PEs. Additionally, each PE has non-coherent, 8 kB, 2-way 326 set associative caches, one for data and another for instructions. Compute 327 clusters can communicate with each other and can indirectly access an 328 external LP-DDR3 of 2 GB through data and control NoCs. Compilation 329 was done using Kalray Compiler 9898489 (based on GCC 4.9) with the 330 flags -03, -fopenmp, and -ffast-math. 331

 Xeon Phi: The processor used in this article features 57 4-way multithreaded cores. Each one of its cores has a clock frequency of 1.10 GHz, 32 kB for L1 instruction and data caches and 512 kB for L2 cache. Every core can directly address the shared DDR memory (6 GB) and is connected to the remaining cores by a high-performance bidirectional ring-shaped NoC. Intel's ICC version 14.0 was used to compile the code with the following compilation flags: -03, -openmp, -mmic (enables the cross compiler nedeed for a native execution on the Xeon Phi), and -fp-model fast=2. Additionally during development, we used the flag -vec-report2 to verify whether the seismic wave simulation kernel was successfully vectorized.

• Tesla K20: This graphics board features NVIDIA Kepler architecture with 2496 CUDA parallel-processing cores, with working frequencies up to 758 MHz and 5 GB of GDDR5 GPU memory. Compilation was done using NVIDIA's nvcc version 5.5 with the flags -O3 -arch sm_20, and --use_fast_math, using GCC 4.9 as the host compiler.

342

343

344

345

346

• Xeon E5: This platform features a Xeon E5-4640 Sandy Bridge-EP processor, which has 8 physical cores, with working frequencies up to 2.4 GHz and 32 GB of DDR3 memory. The code was compiled with GCC 4.9 with the flags -03, -fopenmp, and -ffast-math.

Altix UV 2000: It is a Non-Uniform Memory Access (NUMA) platform composed of 24 NUMA nodes. Each node has a Xeon E5-4640 Sandy Bridge-EP processor (with the same specifications of the Xeon E5 platform) and 32 GB of DDR3 memory shared in a cc-NUMA fashion (NUMAlink6). Overall, this platform has 192 physical cores. Compiler version and flags are identical to that of the Xeon E5 platform.

We use four metrics to compare the energy and computing performance: 357 time-to-solution, energy-to-solution, speedup and Energy-delay Product (EDP). 358 Time-to-solution is the time spent to reach a solution for a seismic wave prop-350 agation simulation. Analogously, energy-to-solution is the amount of energy 360 spent to reach a solution for a seismic wave propagation simulation. Speedups 361 are calculated by dividing the time-to-solution of the sequential version by time-362 to-solution of the parallel/distributed version with n threads. The EDP, ini-363 tially proposed by *Horowitz et al.* [28], fuses the time-to-solution and energy-to-364 solution into a single metric, allowing hardware platforms to be directly com-365 pared taking into account both metrics. Since the relative weights between per-366 formance and energy consumption are subject to pragmatic concerns, we show 367 the values for both EDP and Energy-delay-squared Product (ED^2P) [29]. All 368 results represent averages of 30 runs to guarantee statistically relevant values. 369 The energy-to-solution was obtained through each platform's specific power 370 measurement sensors. Both Xeon E5 and Altix UV 2000 are based on Intel's 371 Sandy Bridge microarchitecture. This microarchitecture has Intel's Running 372 Average Power Limit (RAPL) interface which allows us to measure the power 373 consumption of CPU-level components through hardware counters. We used 374

this approach to obtain the energy consumption of the whole CPU package including cores and cache memory.

On Xeon Phi, we used Intel's MIC System Management and Configuration (MICSMC) tool which allows us to monitor the processor's power consumption.

Power measurements obtained from RAPL and MICSMC are very accurate as 379 shown in [30, 31]. Similarly, MPPA-256 features a tool called K1-POWER to 380 collect energy measurements of the whole chip, including all clusters, on-chip 381 memory, I/O subsystems and NoCs. According to Kalray's reference manuals, 382 the measurement precision on MPPA-256 is ± 0.25 W. NVIDIA Kepler GPUs 383 such as Tesla K20 also have a similar tool called NVIDIA Management Library 384 (NVML). We used the NVML to gather the power usage for the GPU and its as-385 sociated circuitry (e.g., internal memory). According to NVML documentation, 386 readings are accurate to within $\pm 5\%$ of the actual power draw. 387

MPPA-256, Xeon Phi and Tesla K20 architectures can all be used as accelera-388 tors, where the main application code is executed on the host and performance-389 critical sections of the seismic wave propagation kernel are offloaded to them. 390 However, since we intend to compare the performance and energy consumption 391 of our seismic wave propagation solutions for these processors, we execute the 392 entire application on MPPA-256 and Xeon Phi in native mode. In this case, both 393 the main application code and the seismic wave propagation kernel are cross-394 compiled for them. GPUs, however, cannot be used in native mode. For that 395 reason we avoid host intervention whenever possible. During our tests the host 396 was only responsible for loading the kernel and for performing the initial and 397 final data exchanges between the host and GPU memories. To provide a fair 398 comparison and discard influences caused by the host, all the results presented 399 in the following sections reflect only the execution of the kernel. 400

401 *4.2. MPPA-256*

Figure 3(a) shows the impact of the number of prefetched planes on commu-402 403 nication and computation times. As we increase the number of planes available at the compute cluster memory level, we improve the reuse of data in the verti-404 cal direction. This allows us to overlap a considerable portion of data transfers 405 with computations. However, we observed only slight improvements past six 406 planes. This is due to the saturation of the NoC as this strategy increases the 407 data traffic each time we increase the number of prefetched planes. Contrary to 408 the other architectures, the small amount of memory available at each compute 409 cluster on MPPA-256 (2 MB) obliges us to perform an important number of data 410 transfers from/to the DDR. Due to the limited on-chip memory, were able to 411 prefetch only up to eight planes before exhausting the available local memory in 412 each compute cluster. Overall, the reason why the communication is still fairly 413 414 high compared to computation is because the ratio between the time taken by a compute cluster to compute a subdomain of 1.5 MB and the time taken to 415 transfer 1.5 MB from/to the DDR is low. This limits the performance gains that 416 can be achieved by overlapping communications with computations. 417

Figure 3(b) presents a weak scalability analysis of the seismic wave propagation kernel on MPPA-256 when we vary the number of prefetched planes and the number of clusters. For this analysis the problem size assigned to each cluster remains constant as we increase the number of clusters. Therefore, linear scaling is achieved if the normalized execution time is kept constant at 1.00 while the workload is increased in direct proportion to the number of clusters.



Figure 3: Seismic wave propagation simulation kernel on MPPA-256.

Although the increase on the number of prefetched planes can indeed improve 424 the scalability of our solution, there is still a considerable performance degra-425 dation as we employ an additional number of clusters. Even if our prefetching 426 scheme is capable of masking communication costs on MPPA-256, the latency 427 and bandwidth of the NoC can still hurt the execution performance and thus 428 limit the scalability. A comparative scalability evaluation between MPPA-256 429 and Altix UV 2000 is presented in [8]. However, the results presented on that 430 paper are slightly different from the ones we present here. This is due to the use 431 of a newer toolchain in this work which has a slight performance improvement 432 as well as better energy efficiency. 433

434 4.3. Xeon Phi

As discussed in Section 3.3, our solution to seismic wave propagation simulations on Xeon Phi employs not only the cache blocking technique, but also classic memory alignment with pointer shifting. Besides these optimizations, we experimented with different thread affinity and scheduling strategies.

Thread affinity can reduce variations on the execution time of an applica-439 tion and in some cases improve execution times by performing a better thread 440 placement. We executed the seismic wave propagation kernel using the follow-441 ing OpenMP's thread placement policies: compact, scatter, balanced and default 442 (no thread affinity). Setting affinity to compact will place OpenMP threads 443 by filling cores one by one. Balanced affinity evenly distributes threads among 444 the cores. It attempts to use all the available cores while keeping the thread 445 neighboring logical IDs physically close to one another. The scatter affinity also 446 evenly distributes threads among the cores but it does so in a round-robin fash-447 ion. Hence, threads with the adjacent IDs are not guaranteed to be physically 448 adjacent. 449

Scheduling strategies dictate how the loop iterations are assigned to threads.
We experimented with three scheduling strategies available in OpenMP: *static*, *dynamic* and *guided*. In static scheduling, the number of loop iterations is
statically divided by the number of threads. This results in equal-sized chunks
that are assigned to the threads (or as equal as possible in the case where the



Figure 4: Impact of optimizations for dynamic and static OpenMP scheduling policies.

number of loop iterations is not evenly divisible by the number of threads). 455 In dynamic scheduling, blocks of loop iterations are put into a shared work 456 queue. These blocks are then dynamically distributed to the threads. When 457 a thread finishes its block, it retrieves the next block of loop iterations from 458 the top of the work queue. The size of the blocks is fixed and determined by 459 the OpenMP's chunk parameter. The guided scheduling is similar to dynamic 460 scheduling, but the size of the blocks starts off large and decreases to better 461 handle load imbalance between iterations. In this case, the OpenMP's chunk 462 parameter defines the approximate minimum size of the block. 463

Figure 4 shows the speedup we obtained when the kernel was executed 464 with dynamic scheduling with no optimizations (*Baseline*); only memory align-465 ment (MA); only cache blocking (CB); memory alignment and cache blocking 466 (MA+CB). Moreover, it presents the results obtained with static and guided 467 scheduling policies when all optimizations are applied. The chunk size used for 468 the experiments with dynamic and guided scheduling policies was 1, since it 469 presented the best results¹. Thread affinity (Affinity) was only applied to static 470 471 scheduling. As we will explain later on, thread affinity has almost no impact when OpenMP's dynamic or guided scheduling policies are used. 472

Experimental results showed us that memory alignment is essential to make a good use of Xeon Phi's 512-bit vector instructions. However, in order to do so, cache blocking is indispensable. In fact, memory alignment alone accounted for an average improvement of 24 % in comparison to the baseline. The performance improvement of this optimization is severely limited by the poor utilization of the local caches. On the other hand, when compared to the baseline, a more

 $^{^1\}mathrm{Experiments}$ with chunks greater than 1 exhibited higher load imbalances resulting in poorer performance.

efficient utilization of the caches through cache blocking alone increased the 479 performance by 79% on average. If used in isolation, cache blocking does not 480 allow a full use of Xeon Phi's vectorization capabilities and creates additional 481 cache faults due to unaligned memory accesses. By using these two techniques 482 at once we can, at the same time, make a good utilization of Xeon Phi's vectorial 483 capabilities and its fast local caches. Indeed, when these two techniques are 484 used together we see an average improvement of 92% in performance, and an 485 improvement of 91% for 224 threads. 486

Dynamic and guided scheduling policies had a better scaling behavior as well 487 as better performance than static scheduling. On average, dynamic scheduling 488 provided an increase of 14% over static scheduling. Static scheduling has the 489 downside of causing imbalances during the execution of the simulation with an 490 intermediate number of threads. For instance, for 58 threads, iterations would 491 be divided equally between threads, but since a single core would execute two 492 threads, it would actually end up executing twice as many iterations as the other 493 cores. This behavior can clearly be seen as steps in the speedup graph. However, 494 this assumption is not enough to effectively explain why the performance using 495 static scheduling is not on par with (or better than) the performance of dynamic 496 scheduling when the number of threads is a multiple of the number of cores. 497

The reason for that lies on the high usage levels of the NoC during the 498 implicit synchronization between the calculation of the velocity and stress com-499 ponents of the stencil. For each timestep, the stress calculation loop only starts 500 its execution after all the velocity components of the previous loop have finished 501 502 (cf. Algorithm 2.1). This synchronization makes every thread to try to commu-503 nicate at the same time at the start of each parallel loop. On the static scheduler, the concurrency for the NoC might introduce small delays for the beginning of 504 the execution of each thread, essentially making the loop's critical path slightly 505 longer. While by itself each one of these delays is not very significant, they 506 accumulate at each timestep both at the stress and velocity calculation loops. 507 Since the concurrency for the NoC also increases with the number of threads, 508 the relative execution performance difference between the static and dynamic 509 approaches also increases. This imbalance is compensated on the dynamic case, 510 even if the dynamic scheduler has a higher overhead, because it is able to keep 511 more threads active up to the end of each loop. 512

Guided scheduling also allows for a better work distribution and smaller 513 imbalances than the static scheduler between threads at the end of each loop. 514 However, the synchronization between velocity and stress loops and the small 515 delay introduced during the initial phases of each loop still create slight imbal-516 ances at the end of both the stress and velocity loops of each timestep. Using 517 the guided scheduler we were able to achieve near linear speedups up to the 518 number of cores. After this point, the dynamic scheduler gradually approaches 519 the guided performance to eventually beat it with the best execution time with 520 224 threads. 521

Finally, for static scheduling, thread affinity was able to improve the performance of MA + CB in 11%, on average. On the other hand, by its nature, dynamic and guided scheduling reduce the importance of thread affinity. Indeed, scatter and balanced thread placement policies presented very similar results in these cases. However, the compact thread placement policy decreased the performance with dynamic and guided scheduling by 22 % on average, since it creates an important imbalance between cores, specially for low thread-counts. The loss in performance is reduced as the number of threads increases, from 50 % for 40 threads to 0 % for thread counts beyond 222.

Since Xeon Phi does not offer Dynamic Voltage and Frequency Scaling (DVFS),
 the energy consumption on this platform is directly related to the execution time
 of the application. The best execution time was obtained with dynamic schedul ing with 224 threads, providing the smallest energy-to-solution at 4.42 kJ, and

thus resulting in an improvement of 91% when compared to the baseline (46.76 kJ).

536 4.4. Overall Energy and Performance Results

In this section we compare the performance and energy consumption of our seismic wave propagation simulation kernel on MPPA-256 and Xeon Phi against other multicore and GPU reference implementations. On Xeon E5, we used the solution proposed by *Dupros et al.* [26], which employs OpenMP for the parallelization. On GPU we relied on the solution proposed by *Michéa and Komatitsch* [27], which is a state-of-the-art parallel solution for GPU-based seismic wave propagation simulation.

DVFS can make a significant difference in both performance and energy 544 consumption. Although not available on the manycore processors we evaluated, 545 it is available for the Xeon E5 and GPU platforms. Therefore, for these platforms 546 we always show two measurements. The first ones, Xeon E5 (2.4 GHz) and Tesla 547 K20 (758 MHz), represent the experimental results when their frequencies are 548 549 optimized for performance, *i.e.*, using their maximum working frequencies. The second ones, Xeon E5 (1.6 GHz) and Tesla K20 (705 MHz), are relative to the optimal 550 energy consumption setting, which for this kernel was $1.6 \,\mathrm{GHz}$ and $705 \,\mathrm{MHz}$ on 551 Xeon Phi and Tesla K20, respectively. 552

Figure 5 compares the time-to-solution and energy-to-solution across the processors using a problem size of 2 GB (180³ grid points) and 500 time steps. For these experiments we used the optimal number of threads on each platform. With the exception of Xeon Phi (in which the best results were obtained with 224 threads), the thread count was equal to the number of physical cores of each processor. As shown in Figure 4, our solution for Xeon Phi keeps scaling considerably well past the 57 physical cores.

560 To the best of our knowledge, GPUs are among the most energy efficient platforms currently in use for seismic wave propagation simulation. Yet, our 561 proposed solution on MPPA-256 achieves the best energy-to-solution among the 562 analyzed processors, consuming 78%, 77%, 88%, 87% and 86% less energy 563 than Tesla K20 (758 MHz), Tesla K20 (705 MHz), Xeon Phi, Xeon E5 (2.4 GHz), and 564 Xeon E5 (1.6 GHz), respectively. On the other hand, when we consider the time-565 to-solution, MPPA-256 does not have the upper hand. In fact Tesla K20 had the 566 best execution times among all the evaluated architectures. This result was, 567 however, to be expected since the theoretical peak performance of Tesla K20 568 (3.5 TFLOPS in single precision) is at least 75 % higher than that of the second 569



Figure 5: Chip-to-chip comparison.

placed architecture Xeon Phi (2.0 TFLOPS in single precision). Our solution 570 for Xeon Phi improved the execution performance in 58%, 72% and 73% when 571 compared to Xeon E5 (2.4 GHz), Xeon E5 (1.6 GHz) and MPPA-256, respectively. 572 When compared to the Tesla K20 (758 MHz), our solution for Xeon Phi was only 573 $20\,\%$ slower, which not only is compatible to the nominal performance difference 574 between these architectures, but also shows that our solution was able to provide 575 hardware utilization levels that are at least as efficient as a state-of-the-art 576 solution for GPUs. 577

A naïve direct comparison between the raw performance or the average power consumption between Altix UV 2000, a multi-processor architecture, to the previous single-processor architectures would make little sense. Therefore, we compare the Altix UV 2000 NUMA platform to the other processors in terms of energy efficiency and EDP. Table 1 shows the EDP and ED²P for all platforms (the lower the better). A scalability analysis of the seismic wave propagation kernel on Altix UV 2000 is discussed by *Castro et al.* [8].

When we consider EDP, MPPA-256 and Tesla K20 are clearly superior to 585 the other evaluated platforms. If energy-to-solution alone is considered, MPPA-586 256 has a clear advantage. MPPA-256 was conceived as a low-power highly-587 parallel processor. During our evaluation the MPPA-256 consumed on average 588 only 6.45 W, while Tesla K20, Xeon Phi and consumed 184 W, 108 W and Altix UV 589 2000 1458 W, respectively. However Tesla K20 shows a better balance between 590 energy consumption and performance than the other manycore processors as 591 evidenced by ED^2P . The high energy consumption of Xeon Phi places it right 592 between MPPA-256/Tesla K20 and Xeon E5 when EDP is considered. However 593 when we increase the time-to-solution weight (ED^2P) , Xeon Phi becomes more 594 interesting than MPPA-256. 595

The Altix UV 2000 platform has the best trade-off between the time-tosolution and energy-to-solution, for both maximum performance and optimal energy consumption settings. If we consider ED²P, this difference becomes even more noticeable. The significant differences highlighted in the combined time and energy evaluation can be explained by the poor arithmetic intensity

Table 1: EDP (in thousands) for a simulation with 180^3 grid points and 500 time steps.

Platform	EDP	$\mathrm{E}\mathrm{D}^{2}\mathrm{P}$
MPPA-256	49	$4,\!695$
Xeon Phi	114	2,934
Tesla K20 $(758 \mathrm{MHz})$	49	1,056
Tesla K20 $(705 \mathrm{MHz})$	50	1,124
Xeon E5 $(2.4 \mathrm{GHz})$	242	14,755
Xeon E5 (1.6 GHz)	330	30,143
Altix UV 2000 (2.4 GHz)	13	40
Altix UV 2000 $(1.6 \mathrm{GHz})$	17	74

(FLOP/transferred byte ratio) which is characteristic of seismic wave kernels 601 based on elastodynamics stencils. These memory-bound applications can ex-602 perience important losses of performance if communications are not carefully 603 overlapped with communications. On manycores such as MPPA-256 and Xeon 604 Phi, there is an additional overhead created by the limited amount of fast local 605 memory which forces the application to frequently employ the NoC during the 606 execution. Indeed, Altix UV 2000 has approximately 20 times more fast local 607 memory per thread than MPPA-256 and Xeon Phi. This reduces considerably the 608 communication overhead associated to accesses to the main memory. 609

610 5. Related Work

Totoni et al. [6] compared the power and performance of Intel's Single-Chip 611 Cloud Computer (SCC) to other types of CPUs and GPUs. The analysis was 612 based on a set of parallel applications implemented with the Charm++ pro-613 gramming model. Although they showed that there is no single solution that 614 always achieves the best trade-off between power and performance, the results 615 suggest that manycores are an opportunity for the future. Morari et al. [3] 616 proposed an optimized implementation of radix sort for the Tilera TILEPro64 617 manycore processor. The results showed that the their solution for TILEPro64 618 provides much better energy efficiency than an general-purpose multicore pro-619 cessor (Intel Xeon W5590) and comparable energy efficiency with respect to a 620 GPU NVIDIA Tesla C2070. 621

Francesquini et al. [7] evaluated three different classes of applications (CPU-622 bound, memory-bound and mixed) using highly-parallel platforms such as MPPA-623 256 and a 24-node, 192-core NUMA platform. They showed that manycore 624 architectures can be very competitive, even if the application is irregular in 625 nature. Their results showed that MPPA-256 may achieve better performance 626 than a traditional general-purpose multicore processor (Intel Xeon E5-4640) on 627 CPU-bound and mixed workloads whereas on a memory-bound workload Xeon 628 E5 had better performance than MPPA-256. Among the evaluated platforms, 629 MPPA-256 presented the best energy efficiency reducing the energy consumed 630 on cpu-bound, mixed and memory-bound applications by at least 6.9x, 6.5x 631 and 3.8x, respectively. 632

Using the low-power Adapteva Epiphany-IV manycore, Varghese et al. [5] 633 described how a stencil-based solution to the anisotropic heat equation using a 634 two-dimensional grid was developed. This manycore has a low power budged 635 (2W) and has 64 processing cores. Each core has a local scratchpad of 32 kB and 636 no cache-memory. Cores are connected by a NoC which allows direct memory 637 access to 1 GB of RAM and also to other cores' local memories. The challenges 638 we faced for the development of our seismic wave propagation simulation on the 639 MPPA-256 architecture are similar to the ones faced by the authors of this work. 640 Similar to MPPA-256, Epiphany-IV has a very limited amount of local memory 641 available to each core and no automatic prefetching mechanism exists; every 642 data movement has to be explicitly controlled by the application. The authors 643 demonstrated that, even if the implementation of an efficient application can 644 be a challenging task, their solution was able to achieve a FLOPS/Watt ratio 3645 times better than an Intel 80-core Terascale Processor. 646

Adapting seismic wave propagation numerical kernels to emerging parallel architectures is an active research topic. Due the versatility of the FDM used both for oil and gas applications and standard earthquakes modeling, several strategies have been proposed. For instance, Intel and NVIDIA have proposed optimized implementations of such stencils [25, 32].

Dupros et al. [33] presented a review of the scalability issues for distributed 652 and shared-memory platforms with a focus on mapping processes/threads on 653 hierarchical clusters. Dursun et al. [34] introduced several additional strategies, 654 including inter-node and intra-node optimizations. Recently, Christen et al. [35] 655 656 described the use of the Patus framework to optimize the AWP-ODC finite difference code. In particular, they detailed the impact of vectorization and 657 cache prefetching and reported a performance improvement up to 15% when 658 running the complete application with 512 MPI processes. 659

Several research efforts have been done on the adaptation of seismic wave ker-660 nels on GPUs to overcome the poor arithmetic intensity (FLOPS/transferred 661 byte ratio) offered by elastodynamics stencils [27, 36]. The results obtained 662 demonstrate that GPUs could be a relevant alternative to standard general-663 purpose processors. Krueger et al. [37] compared the performance and energy 664 efficiency of an Intel Nehalem platform, an NVIDIA Tesla GPU and a simulated 665 general-purpose manycore chip design optimized for high-order wave equations 666 called "Green Wave". They showed that Green Wave can be up to 8x and 3.5x 667 more energy efficient per node when compared with the Nehalem and GPU 668 platforms, respectively. Differently from this work, our experiments and mea-669 surements were carried out on real manycore processors. Even if the low-power 670 MPPA-256 is not optimized for high-order wave equations as Green Wave, we 671 were still able to achieve energy efficiency improvements of at least 86 % and 672 77% when compared to the Xeon E5 and Tesla K20, respectively. 673

674 6. Conclusion and Perspectives

The use of highly-parallel manycore processors to meet the high demand for data processing capabilities needed by parallel scientific simulations has recently become one of central points of interest in the HPC community. In this paper,
we presented our approach to the seismic wave propagation simulation using the
MPPA-256 and Xeon Phi manycore processors. Although both processors share
some architectural characteristics, they present different challenges that must
be overcome in order to obtain good performance results.

The low-power MPPA-256 manycore processor has a limited amount of dis-682 tributed low-latency memories that must be explicitly managed by the program-683 mer. Xeon Phi, on the other hand, is a performance-centric manycore processor 684 that requires careful source code changes to help the compiler to effectively vec-685 torize time-consuming loops and to improve cache locality. In this article we 686 proposed a new multi-level tiling strategy and a software prefetching mechanism 687 that allowed us to perform real-world seismic wave propagation simulations and 688 at the same time to lighten the communication overhead imposed by the NoC 689 on the MPPA-256. On the other hand, on the Xeon Phi architecture we employed 690 a cache blocking technique along with memory alignment and thread affinity 691 strategies which allowed us to considerably improve the performance and scal-692 ability of our solution. 693

Our results showed that our approach to the MPPA-256 is the most energy 694 efficient whereas our solution for the Xeon Phi achieves a performance compara-695 ble to the state-of-the-art solution for GPUs. MPPA-256 presented energy con-696 sumption improvements of 77%, 86% and 88% over other solutions for GPU, 697 general-purpose multicore and the Xeon Phi manycore architectures, respectively. 698 In terms of performance, on the other hand, our solution for Xeon Phi achieved 699 performance improvements of 73%, 58% with respect to the solutions for MPPA-700 701 256 and general-purpose multicore processor, respectively.

Despite encouraging results, our solutions for MPPA-256 and Xeon Phi many-702 core processors could still be improved. On MPPA-256, the communication still 703 consumes a large amount of the total execution time (58%). This is due to 704 the high traffic on the NoC to access a single DDR memory. Kalray recently 705 announced a multi-MPPA solution that features four MPPA-256 processors on 706 the same board with less than 50 W of power consumption. In this new solution, 707 each MPPA-256 processor can access two DDR3 channels in parallel and MPPA-708 256 processors are interconnected through NoC eXpress interfaces (NoCX), pro-709 viding an aggregate bandwidth of $40 \, \text{GB/s}$. The benefits of this new solution 710 for seismic wave propagation simulations are two-fold: (i) this would allow us 711 to deal with input problem sizes of 32 GB or more; and (ii) distributing data 712 among different MPPA-256 processors along with the parallel access to two DDR3 713 memories in each processor would alleviate the current communication bottle-714 neck. Thus, we plan to work on new versions of our multi-level tiling strategy 715 and prefetching scheme to exploit the full potential of multi-MPPA solutions as 716 soon as they become available. 717

On Xeon Phi, the performance could still be improved by using an auto-tuning mechanism to automatically select the best size of the tiles and specific optimizations from the compiler based on the input problem and the number of threads used. Thus, we intend to study if frameworks such as *Pochoir* [38] or *Patus* [35] could provide us the essentials to implement the auto-tuning approach.

Finally, we intend to study how our optimizations proposed in this paper 723 could be made automatic. One possibility would be to extend PSkel [39], a state-724 of-the-art framework that provides a single high-level abstraction for stencil 725 programming, to support MPPA-256 and Xeon Phi manycores. This would allow 726 us to implement our proposed optimizations inside the framework, so applica-727 tions implemented in PSkel will automatically benefit from software prefetching 728 (MPPA-256) and cache blocking, memory alignment and thread affinity (Xeon 729 Phi). 730

731 Acknowledgments

This work was supported by FAPESP grant 2014/17925-0, PNPD/CAPES, the European Community's Seventh Framework Programme (FP7/2007-2013) under the Mont-blanc 2 Project (www.montblanc-project.eu) grant 610402, and BRGM Carnot-institute. We would also like to thank Bull and the Center for Excellence in Parallel Programing (CEPP) for granting us access to their hardware infrastructure which was used in our experiments with Xeon Phi.

738 **References**

- [1] N. Rajovic, N. Puzovic, L. Vilanova, C. Villavieja, A. Ramirez, The low-power architecture approach towards exascale computing, in: Workshop on Scalable Algorithms for Large-scale Systems (ScalA), ACM, Seattle, USA, 2011, pp. 1–2.
- [2] D. Göddeke, D. Komatitsch, M. Geveler, D. Ribbrock, N. Rajovic, N. Pu-zovic, A. Ramirez, Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster, Journal of Computational Physics 237 (2013) 132–150.
- A. Morari, A. Tumeo, O. Villa, S. Secchi, M. Valero, Efficient sorting on the Tilera manycore architecture, in: International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), IEEE Computer Society, New York, USA, 2012, pp. 171–178.
- [4] B. D. de Dinechin, P. G. de Massasa, G. Lagera *et al.*, A distributed runtime environment for the Kalray MPPA-256 integrated manycore processor,
 in: International Conference on Computational Science (ICCS), Elsevier,
 Barcelona, Spain, 2013, pp. 1654–1663.
- [5] A. Varghese, B. Edwards, G. Mitra, A. P. Rendell, Programming the Adapteva Epiphany 64-core network-on-chip coprocessor, in: International Parallel Distributed Processing Symposium Workshops (IPDPSW), IEEE Computer Society, Phoenix, USA, 2014, pp. 984–992.
- [6] E. Totoni, B. Behzad, S. Ghike, J. Torrellas, Comparing the power and performance of Intel's SCC to state-of-the-art CPUs and GPUs, in: International Symposium on Performance Analysis of Systems and Software (ISPASS), IEEE Computer Society, New Brunswick, Canada, 2012, pp. 78–87.

- [7] E. Francesquini, M. Castro, , P. H. Penna, F. Dupros, H. C. de Freitas, P. O. A. Navaux, J.-F. Méhaut, On the energy efficiency and performance of irregular application executions on multicore, NUMA and manycore platforms, Journal of Parallel and Distributed Computing, In Press. doi:10.1016/j.jpdc.2014.11.002.
- [8] M. Castro, F. Dupros, E. Francesquini, J.-F. Méhaut, P. O. A. Navaux, Energy efficient seismic wave propagation simulation on a low-power many-core processor, in: International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), IEEE Computer Society, Paris, France, 2014, pp. 57–64.
- [9] E. Francesquini, A. Goldman, J.-F. Mehaut, A NUMA-Aware Runtime
 Environment for the Actor Model, in: International Conference on Parallel
 Processing (ICPP), IEEE, Lyon, France, 2013, pp. 250–259.
- [10] L. L. Pilla, C. P. Ribeiro, D. Cordeiro, C. Mei, A. Bhatele, P. O. Navaux,
 F. Broquedis, J.-F. Mehaut, L. V. Kale, A hierarchical approach for load
 balancing on parallel multi-core systems, Proceedings of the 41st International Conference on Parallel Processing 0 (2012) 118–127.
- [11] M.J. Rashti *et al.*, Multi-core and network aware MPI topology functions,
 in: Proceedings of the 18th European MPI Users' Group conference on
 Recent advances in the message passing interface, EuroMPI'11, 2011, pp.
 50–60.
- [12] G. Mercier, J. Clet-Ortega, Towards an efficient process placement policy for mpi applications in multicore environments, in: Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer, 2009, pp. 104–115.
- [13] J. McCalpin, Paleoseismology, 2nd Edition, Vol. 95, Academic press, 2009,
 Appendix 1.
- [14] J. Lysmer, L. Drake, A finite element method for seismology, Methods in
 Computational Physics 11 (1972) 181–216.
- [15] D. Komatitsch, J. P. Vilotte, The spectral-element method: An efficient tool to simulate the seismic response of 2D and 3D geological structures, Bulletin of the Seismological Society of America 88 (2) (1998) 368–392.
- [16] M. Dumbser, M. Käser, An arbitrary high-order discontinuous Galerkin
 method for elastic waves on unstructured meshes II: The three-dimensional
 isotropic case, Geophys. J. Int. 167 (1) (2006) 319–336.
- ⁷⁹⁹ [17] J. Virieux, *P-SV* wave propagation in heterogeneous media: Velocity-stress ⁸⁰⁰ finite-difference method, Geophysics 51 (1986) 889–901.
- [18] E. H. Saenger, N. Gold, S. A. Shapiro, Modeling the propagation of elastic
 waves using a modified finite-difference grid, Wave Motion 31 (1999) 77–92.
- [19] W. Zhang, Z. Zhang, X. Chen, Three-dimensional elastic wave numerical modeling in the presence of surface topography by a collocated-grid finitedifference method on curvilinear grids, Geophysical Journal International 190 (1) (2012) 358–378.
- ⁸⁰⁷ [20] H. Aochi, T. Ulrich, A. Ducellier, F. Dupros, D. Michea, Finite difference simulations of seismic wave propagation for understanding earthquake

- physics and predicting ground motions: Advances and challenges, Journal
 of Physics: Conference Series 454 (2013) 1–9.
- [21] P. Aubry, P.-E. Beaucamps *et al.*, Extended cyclostatic dataflow program
 compilation and execution for an integrated manycore processor, in: Procedia Computer Science, Vol. 18, Elsevier, Barcelona, Spain, 2013, Ch. 2013
 International Conference on Computational Science, pp. 1624–1633.
- [22] K. Datta, S. Kamil, S. Williams, L. Oliker, J. Shalf, K. Yelick, Optimization and performance modeling of stencil computations on modern microprocessors, SIAM Review 51 (1) (2009) 129–159.
- [23] M. Bianco, B. Cumming, A generic strategy for multi-stage stencils, in: International Conference on Parallel and Distributed Computing (Euro-Par), ACM, Porto, Portugal, 2014, pp. 584–595.
- [24] G. Rivera, C.-W. Tseng, Tiling optimizations for 3D scientific computations, in: Supercomputing Conferece (SC), ACM, Dallas, USA, 2000, pp. 32–38.
- [25] J. Reinders, J. Jeffers, High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches, 1st Edition, Morgan Kaufmann, 2014.
- [26] F. Dupros, C. Pousa, A. Carissimi, J.-F. Méhaut, Parallel simulations of seismic wave propagation on NUMA architectures, in: International Conference on Parallel Computing (ParCo), IOS Press, Lyon, France, 2010, pp. 67–74.
- [27] D. Michéa, D. Komatitsch, Accelerating a three-dimensional finitedifference wave propagation code using GPU graphics cards, Geophysical Journal International 182 (1) (2010) 389–402.
- [28] M. Horowitz, T. Indermaur, R. Gonzalez, Low-power digital design, in: Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium, 1994, pp. 8–11. doi:10.1109/LPE.1994.573184.
- [29] A. J. Martin, Towards an energy complexity of computation, Information Processing Letters 77 (2–4) (2001) 181 – 187. doi:http://dx.doi.org/ 10.1016/S0020-0190(00)00214-3.
- [30] M. Hähnel, B. Döbel, M. Völp, H. Härtig, Measuring energy consumption for short code paths using RAPL, ACM SIGMETRICS Performance
 Evaluation Review 40 (3) (2012) 13–17.
- [31] G. Lawson, M. Sosonkina, Y. Shen, Energy evaluation for applications with
 different thread affinities on the Intel Xeon Phi, in: Workshop on Appli cations for Multi-Core Architectures (WAMCA), IEEE Computer Society,
- ⁸⁴⁶ 2014, pp. 54–59.
- [32] P. Micikevicius, 3D finite-difference computation on GPUs using CUDA, in:
 Workshop on General Purpose Processing on Graphics Processing Units,
 ACM, Washington, USA, 2009, pp. 79–84.
- [33] F. Dupros, H.-T. Do, H. Aochi, On scalability issues of the elastodynamics
 equations on multicore platforms, in: International Conference on Computational Science (ICCS), Barcelona, Spain, 2013, pp. 1226–1234.
- 853 [34] H. Dursun, K. ichi Nomura, L. Peng et al., A multilevel parallelization

854	framework for high-order stencil computations, in: International Confer-
855	ence on Parallel and Distributed Computing (Euro-Par), ACM, Delft, The
856	Netherlands, 2009, pp. 642–653.

- [35] M. Christen, O. Schenk, Y. Cui, PATUS for convenient high-performance
 stencils: evaluation in earthquake simulations, in: Supercomputing Con ferece (SC), IEEE Computer Society, 2012, pp. 11:1–11:10.
- [36] R. Abdelkhalek, H. Calandra, O. Coulaud, G. Latu, J. Roman, Fast seismic
 modeling and reverse time migration on a graphics processing unit cluster,
- Concurrency and Computation: Practice and Experience 24 (7) (2012) 739–750.
- ⁸⁶⁴ [37] J. Krueger, D. Donofrio, J. Shalf, M. Mohiyuddin, S. Williams, L. Oliker,
 ⁸⁶⁵ F.-J. Pfreund, Hardware/software co-design for energy-efficient seismic
 ⁸⁶⁶ modeling, in: Supercomputing Conferece (SC), IEEE Computer Society,
 ⁸⁶⁷ 2011, pp. 73:1–73:12.
- ⁸⁶⁸ [38] Y. Tang, R. A. Chowdhury, B. C. Kuszmaul, C.-K. Luk, C. E. Leiser⁸⁶⁹ son, The pochoir stencil compiler, in: ACM Symposium on Parallelism in
 ⁸⁷⁰ Algorithms and Architectures (SPAA), ACM, San Jose, USA, 2011, pp.
 ⁸⁷¹ 117–128.
- [39] A. D. Pereira, L. Ramos, L. F. W. Góes, PSkel: A stencil programming
 framework for CPU-GPU systems, Concurrency and Computation: Practice and Experience.doi:10.1002/cpe.3479.