



HAL
open science

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*, 2016, 10.1016/j.parco.2016.01.011 . hal-01273153

HAL Id: hal-01273153

<https://hal.science/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

^a*Department of Informatics and Statistics, Federal University of Santa Catarina (UFSC)
Campus Reitor João David Ferreira Lima, Trindade, 88040-970, Florianópolis, Brazil*

^b*Institute of Computing, University of Campinas (UNICAMP)
Av. Albert Einstein, 1251, Cidade Universitária, 13083-852, Campinas, Brazil*

^c*BRGM
BP 6009, 45060 Orléans Cedex 2, France*

^d*Informatics Institute, Federal University of Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500, Campus do Vale, 91501-970, Porto Alegre, Brazil*

^e*CEA-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, energy-efficient 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

Email addresses: marcio.castro@ufsc.br (Márcio Castro),
francesquini@ic.unicamp.br (Emilio Francesquini), f.dupros@brgm.fr (Fabrice Dupros),

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 in a reasonable amount of time.

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

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

While some manycore processors may present better energy efficiency than state-of-the-art general-purpose multicore processors [7], their particular architectural characteristics make the development of efficient scientific parallel applications a challenging task [5, 8]. Some of these processors are built and optimized for certain classes of embedded applications like signal processing, video decoding and routing. Additionally, processors such as MPPA-256 have important memory constraints, *e.g.*, limited amount of directly addressable memory and absence of cache coherence protocols. Furthermore, communication costs between cores are not uniform, they depend on the location of the communicating cores. One of the main reasons for this difference in communication cost is the Network-on-Chip (NoC). When applications distribute work among the cores of the processor, they should take into consideration the placement of these tasks and their communication patterns to minimize communication costs. This problem is similar to that faced by applications running on NUMA platforms [9, 10], or MPI applications that try to take into account both the network topology and the memory hierarchy to improve performance [11, 12].

In this paper we outline the architectural distinctiveness of the low-power MPPA-256 manycore processor. Considering these characteristics, we describe 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)

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

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

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

78 The remainder of this paper is organized as follows. Section 2 discusses
79 the fundamentals of seismic wave propagation. Then, Section 3 discusses the
80 main challenges we must overcome when dealing with parallel seismic wave
81 propagations on MPPA-256 and Xeon Phi. Moreover, it presents our approaches
82 to perform efficient seismic wave propagation simulations on both processors.
83 Section 4 discusses performance and energy efficiency experimental results. Sec-
84 tion 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

88 events, such as the $M_W7.9$ (moment magnitude) earthquake in China (2008) or
89 the $M_W8.8$ earthquake in Chile (2010), have been studied using numerical tools
90 with good agreement with observation. From a physical point a view, the mech-
91 anisms used to describe earthquake events are complex and highly nonlinear.
92 For the sake of simplicity, wave propagation simulations usually consider elastic
93 domains and include some anelasticity to account for attenuations. However,
94 in the case of strong motion, these simplifying assumptions are not capable of
95 correctly describing the phenomenon. A fundamental challenge for earthquake
96 engineering is predicting the level and variability of strong ground motion from
97 future earthquakes. Enhancing this predictive ability requires understanding of
98 the earthquake source, the effects of the propagation path on the seismic waves,
99 and basin and near-surface site effects.

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

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

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

126 *2.1. Discretization and Standard Sequential Algorithm*

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

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

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

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

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 $i = j$
 139 and zero otherwise.

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

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

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

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

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

151 For instance, the stencil applied for the computation of the velocity compo-
 152 nent in the x -direction is given by:

$$\begin{aligned} v_x^{(i+\frac{1}{2})jk} \left(l + \frac{1}{2} \right) &= v_x^{(i+\frac{1}{2})jk} \left(l - \frac{1}{2} \right) + a_1 F_x^{(i+\frac{1}{2})jk} \\ &+ a_2 \left[\frac{\sigma_{xx}^{(i+1)jk} - \sigma_{xx}^{ijk}}{\Delta x} + \frac{\sigma_{xy}^{(i+\frac{1}{2})(j+\frac{1}{2})k} - \sigma_{xy}^{(i+\frac{1}{2})(j-\frac{1}{2})k}}{\Delta y} + \frac{\sigma_{xz}^{(i+\frac{1}{2})j(k+\frac{1}{2})} - \sigma_{xz}^{(i+\frac{1}{2})j(k-\frac{1}{2})}}{\Delta z} \right] \\ &- a_3 \left[\frac{\sigma_{xx}^{(i+2)jk} - \sigma_{xx}^{(i-1)jk}}{\Delta x} + \frac{\sigma_{xy}^{(i+\frac{1}{2})(j+\frac{3}{2})k} - \sigma_{xy}^{(i+\frac{1}{2})(j-\frac{3}{2})k}}{\Delta y} + \frac{\sigma_{xz}^{(i+\frac{1}{2})j(k+\frac{3}{2})} - \sigma_{xz}^{(i+\frac{1}{2})j(k-\frac{3}{2})}}{\Delta z} \right] \end{aligned} \quad (5)$$

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

Algorithm 2.1: SEQUENTIAL SEISMIC WAVE PROPAGATION(σ, v)

```

for  $x \leftarrow 1$  to  $x\_dimension$ 
  do { for  $y \leftarrow 1$  to  $y\_dimension$ 
      do { for  $z \leftarrow 1$  to  $z\_dimension$ 
          do { COMPUTE_VELOCITY( $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}$ )
        }
      }
    }
for  $x \leftarrow 1$  to  $x\_dimension$ 
  do { for  $y \leftarrow 1$  to  $y\_dimension$ 
      do { for  $z \leftarrow 1$  to  $z\_dimension$ 
          do { COMPUTE_STRESS( $v_x, v_y, v_z$ )
        }
      }
    }

```

157 The best parallelization strategy for the elastodynamics equations strongly
 158 depends on the characteristics of the underlying hardware architecture. In the
 159 following section, we detail the parallelization and optimization strategies we
 160 employed for the adaptation of this sequential algorithm to highly parallel many-
 161 core processors.

162 **3. Elastodynamics Numerical Kernel on Manycores**

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

169 *3.1. Platforms and Challenges*

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

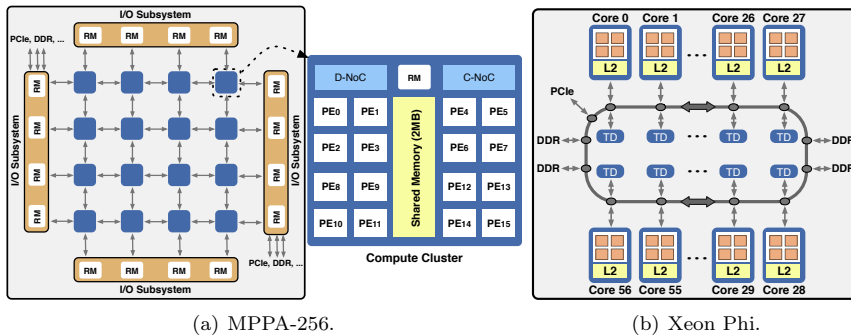


Figure 1: Manycore architectures.

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

185 The development of a seismic wave propagation kernel for this processor
 186 can be a challenging task due to some of its intrinsic characteristics. First, the
 187 low-latency memory available in each compute cluster acts as a cache, whose
 188 goal is to store data retrieved from the DDR. However, data transfers between
 189 the DDR and compute clusters' low-latency memories must be explicitly man-
 190 aged by the programmer, in other words, there are no automatic fetching or
 191 prefetching mechanisms. Second, the amount of data needed to simulate real
 192 wave propagation scenarios does not fit into the local memories. In fact, about
 193 500 kB of memory is always in use by the operating system leaving about 1.5 MB
 194 free to house the application's data and code. For this reason, the application
 195 has to take into account the data transfers and how these data should be sliced
 196 in order to fit into the limited local memory.

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

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

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

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

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

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

240 The three-dimensional structures corresponding to the velocity and stress
 241 fields are allocated on the DDR connected to the I/O subsystem to maximize
 242 the overall problem size that can be simulated. Next, we divide the global
 243 computational domain into several subdomains corresponding to the number
 244 of compute clusters involved in the computation (Figure 2-**1**). This decom-
 245 position provides a first level of data-parallelism. To respect the width of the
 246 stencil (fourth-order), we maintain an overlap of two grid points in each di-
 247 rection. These regions, called ghost zones, are updated at each stage of the
 248 computation with point-to-point communications between neighboring clusters.

249 Unfortunately, only this first level of decomposition is not enough because
 250 the three-dimensional tiles do not fit into the 2 MB of low-latency memories
 251 available to the compute clusters. A second level of decomposition is there-
 252 fore required. We performed this decomposition along the vertical direction as
 253 we tile each three-dimensional subdomain into 2D slices (Figure 2-**2**). This
 254 leads to a significant reduction of the memory consumption for each cluster but
 255 requires maintaining a good balance between the computation and communica-
 256 tion. Our solution relies on a sliding window algorithm that traverses the 3D
 257 domains using 2D planes and overlaps data transfers with computations. This
 258 mechanism can be seen as an *explicit prefetching mechanism* (Figure 2-**3**) as

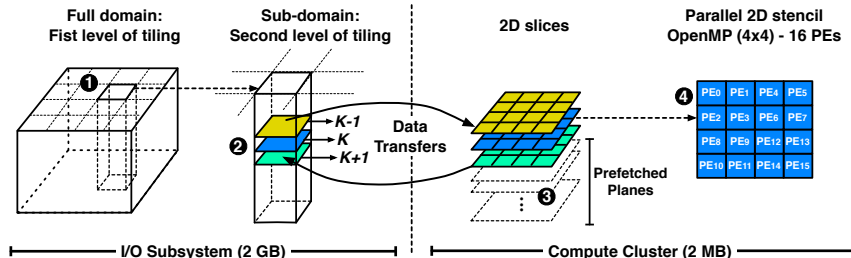


Figure 2: Two-level tiling scheme to exploit the memory hierarchy of MPPA-256.

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

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

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

272 Several standard sequential implementations offer a poor cache reuse and
 273 therefore achieve only a fraction of the peak performance of the processor. Even
 274 if some optimizations alleviate this problem, the standard simulation algorithm
 275 typically scans an array spanning several times the size of the cache using each
 276 retrieved grid point only for a few operations. Therefore, the cost to bring the
 277 needed data from the main memory to the fast local cache memories account for
 278 an important share of the total simulation time, specially for a 3D problem.

279 In particular, this limitation is more noticeable in processors that feature
 280 high core counts such as Xeon Phi due to the contention on the bidirectional
 281 data ring. To improve the cache locality we employ a cache blocking technique.
 282 Cache blocking is an optimization technique that intends to reduce memory
 283 bandwidth bottlenecks. The main idea is to exploit the inherent data reuse
 284 available in the triple nested loop of the elastodynamics kernel by ensuring that
 285 data remains in cache across multiple uses.

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

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

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

309 4. Experimental Results

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

317 4.1. Methodology

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

- 323 • **MPPA-256:** This low-power manycore processor integrates 16 compute
 324 clusters in a single chip running at 400 MHz. Each compute cluster has
 325 16 cores called PEs and a low-latency shared memory of 2 MB shared
 326 between all the PEs. Additionally, each PE has non-coherent, 8 kB, 2-way
 327 set associative caches, one for data and another for instructions. Compute
 328 clusters can communicate with each other and can indirectly access an
 329 external LP-DDR3 of 2 GB through data and control NoCs. Compilation
 330 was done using Kalray Compiler 9898489 (based on GCC 4.9) with the
 331 flags `-O3`, `-fopenmp`, and `-ffast-math`.
- 332 • **Xeon Phi:** The processor used in this article features 57 4-way multi-
 333 threaded cores. Each one of its cores has a clock frequency of 1.10 GHz,
 334 32 kB for L1 instruction and data caches and 512 kB for L2 cache. Every

335 core can directly address the shared DDR memory (6 GB) and is connected
336 to the remaining cores by a high-performance bidirectional ring-shaped
337 NoC. Intel’s ICC version 14.0 was used to compile the code with the fol-
338 lowing compilation flags: `-O3`, `-openmp`, `-mmic` (enables the cross compiler
339 needed for a native execution on the Xeon Phi), and `-fp-model fast=2`.
340 Additionally during development, we used the flag `-vec-report2` to verify
341 whether the seismic wave simulation kernel was successfully vectorized.

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

347 • **Xeon E5:** This platform features a Xeon E5-4640 Sandy Bridge-EP pro-
348 cessor, which has 8 physical cores, with working frequencies up to 2.4 GHz
349 and 32 GB of DDR3 memory. The code was compiled with GCC 4.9 with
350 the flags `-O3`, `-fopenmp`, and `-ffast-math`.

351 • **Altix UV 2000:** It is a Non-Uniform Memory Access (NUMA) platform
352 composed of 24 NUMA nodes. Each node has a Xeon E5-4640 Sandy
353 Bridge-EP processor (with the same specifications of the Xeon E5 platform)
354 and 32 GB of DDR3 memory shared in a cc-NUMA fashion (NUMALink6).
355 Overall, this platform has 192 physical cores. Compiler version and flags
356 are identical to that of the Xeon E5 platform.

357 We use four metrics to compare the energy and computing performance:
358 *time-to-solution*, *energy-to-solution*, *speedup* and *Energy-delay Product (EDP)*.
359 Time-to-solution is the time spent to reach a solution for a seismic wave prop-
360 agation simulation. Analogously, energy-to-solution is the amount of energy
361 spent to reach a solution for a seismic wave propagation simulation. Speedups
362 are calculated by dividing the time-to-solution of the sequential version by time-
363 to-solution of the parallel/distributed version with n threads. The EDP, ini-
364 tially proposed by *Horowitz et al.* [28], fuses the time-to-solution and energy-to-
365 solution into a single metric, allowing hardware platforms to be directly com-
366 pared taking into account both metrics. Since the relative weights between per-
367 formance and energy consumption are subject to pragmatic concerns, we show
368 the values for both EDP and Energy-delay-squared Product (ED²P) [29]. All
369 results represent averages of 30 runs to guarantee statistically relevant values.

370 The energy-to-solution was obtained through each platform’s specific power
371 measurement sensors. Both Xeon E5 and Altix UV 2000 are based on Intel’s
372 Sandy Bridge microarchitecture. This microarchitecture has Intel’s Running
373 Average Power Limit (RAPL) interface which allows us to measure the power
374 consumption of CPU-level components through hardware counters. We used
375 this approach to obtain the energy consumption of the whole CPU package
376 including cores and cache memory.

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

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

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

401 4.2. MPPA-256

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

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

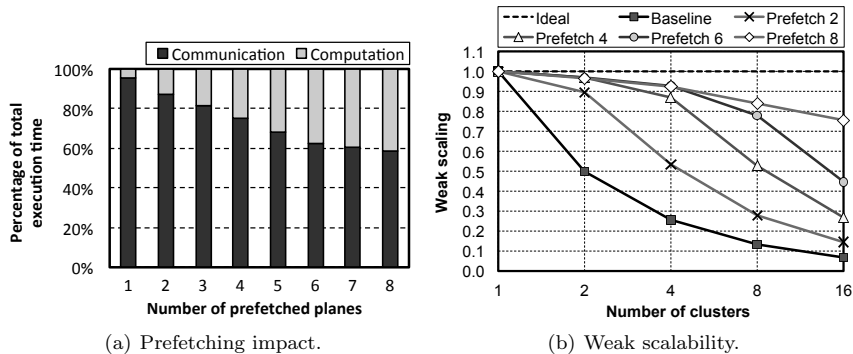


Figure 3: Seismic wave propagation simulation kernel on MPPA-256.

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

4.3. Xeon Phi

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

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

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

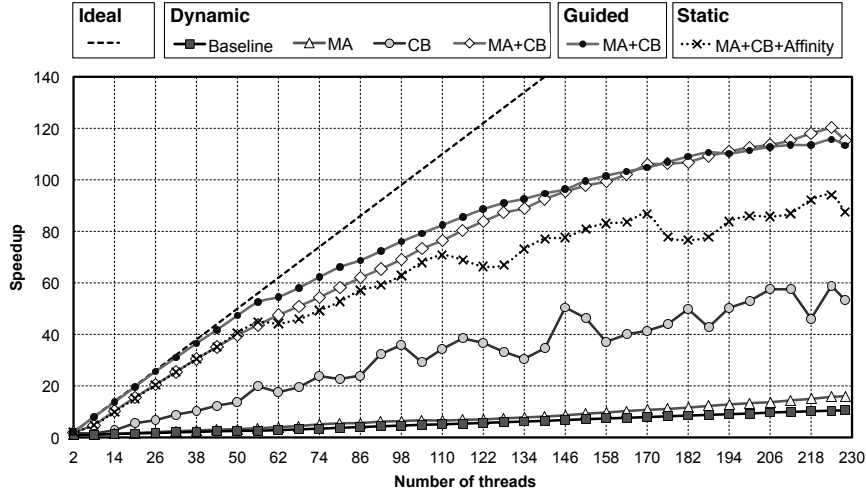


Figure 4: Impact of optimizations for dynamic and static OpenMP scheduling policies.

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

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

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

¹Experiments with chunks greater than 1 exhibited higher load imbalances resulting in poorer performance.

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

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

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

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

522 Finally, for static scheduling, thread affinity was able to improve the perfor-
523 mance of $MA+CB$ in 11 %, on average. On the other hand, by its nature, dy-
524 namic and guided scheduling reduce the importance of thread affinity. Indeed,

525 scatter and balanced thread placement policies presented very similar results
526 in these cases. However, the compact thread placement policy decreased the
527 performance with dynamic and guided scheduling by 22 % on average, since it
528 creates an important imbalance between cores, specially for low thread-counts.
529 The loss in performance is reduced as the number of threads increases, from
530 50 % for 40 threads to 0 % for thread counts beyond 222.

531 Since Xeon Phi does not offer Dynamic Voltage and Frequency Scaling (DVFS),
532 the energy consumption on this platform is directly related to the execution time
533 of the application. The best execution time was obtained with dynamic schedul-
534 ing with 224 threads, providing the smallest energy-to-solution at 4.42 kJ, and
535 thus resulting in an improvement of 91 % when compared to the baseline (46.76 kJ).

536 4.4. Overall Energy and Performance Results

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

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

553 Figure 5 compares the time-to-solution and energy-to-solution across the
554 processors using a problem size of 2 GB (180^3 grid points) and 500 time steps.
555 For these experiments we used the optimal number of threads on each platform.
556 With the exception of Xeon Phi (in which the best results were obtained with
557 224 threads), the thread count was equal to the number of physical cores of
558 each processor. As shown in Figure 4, our solution for Xeon Phi keeps scaling
559 considerably well past the 57 physical cores.

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

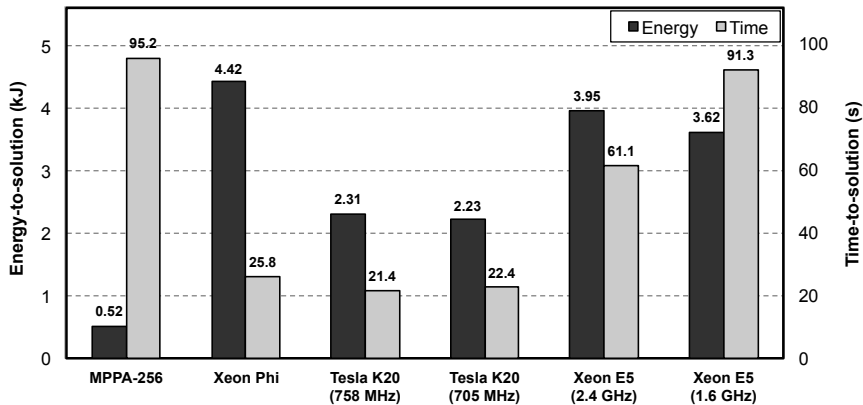


Figure 5: Chip-to-chip comparison.

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

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

585 When we consider EDP, MPPA-256 and Tesla K20 are clearly superior to
 586 the other evaluated platforms. If energy-to-solution alone is considered, MPPA-
 587 256 has a clear advantage. MPPA-256 was conceived as a low-power highly-
 588 parallel processor. During our evaluation the MPPA-256 consumed on average
 589 only 6.45 W, while Tesla K20, Xeon Phi and consumed 184 W, 108 W and Altix UV
 590 2000 1458 W, respectively. However Tesla K20 shows a better balance between
 591 energy consumption and performance than the other manycore processors as
 592 evidenced by ED²P. The high energy consumption of Xeon Phi places it right
 593 between MPPA-256/Tesla K20 and Xeon E5 when EDP is considered. However
 594 when we increase the time-to-solution weight (ED²P), Xeon Phi becomes more
 595 interesting than MPPA-256.

596 The Altix UV 2000 platform has the best trade-off between the time-to-
 597 solution and energy-to-solution, for both maximum performance and optimal
 598 energy consumption settings. If we consider ED²P, this difference becomes
 599 even more noticeable. The significant differences highlighted in the combined
 600 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	ED ² P
MPPA-256	49	4,695
Xeon Phi	114	2,934
Tesla K20 (758 MHz)	49	1,056
Tesla K20 (705 MHz)	50	1,124
Xeon E5 (2.4 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 GHz)	17	74

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

610 5. Related Work

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

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

633 Using the low-power Adapteva Epiphany-IV manycore, *Varghese et al.* [5]
634 described how a stencil-based solution to the anisotropic heat equation using a
635 two-dimensional grid was developed. This manycore has a low power budget
636 (2 W) and has 64 processing cores. Each core has a local scratchpad of 32 kB and
637 no cache-memory. Cores are connected by a NoC which allows direct memory
638 access to 1 GB of RAM and also to other cores' local memories. The challenges
639 we faced for the development of our seismic wave propagation simulation on the
640 MPPA-256 architecture are similar to the ones faced by the authors of this work.
641 Similar to MPPA-256, Epiphany-IV has a very limited amount of local memory
642 available to each core and no automatic prefetching mechanism exists; every
643 data movement has to be explicitly controlled by the application. The authors
644 demonstrated that, even if the implementation of an efficient application can
645 be a challenging task, their solution was able to achieve a FLOPS/Watt ratio 3
646 times better than an Intel 80-core Terascale Processor.

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

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

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

674 6. Conclusion and Perspectives

675 The use of highly-parallel manycore processors to meet the high demand for
676 data processing capabilities needed by parallel scientific simulations has recently

677 become one of central points of interest in the HPC community. In this paper,
678 we presented our approach to the seismic wave propagation simulation using the
679 MPPA-256 and Xeon Phi manycore processors. Although both processors share
680 some architectural characteristics, they present different challenges that must
681 be overcome in order to obtain good performance results.

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

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

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

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

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

731 Acknowledgments

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

738 References

- 739 [1] N. Rajovic, N. Puzovic, L. Vilanova, C. Villavieja, A. Ramirez, The low-
740 power architecture approach towards exascale computing, in: Workshop on
741 Scalable Algorithms for Large-scale Systems (ScalA), ACM, Seattle, USA,
742 2011, pp. 1–2.
- 743 [2] D. Goddeke, D. Komatitsch, M. Geveler, D. Ribbrock, N. Rajovic, N. Pu-
744 zovic, A. Ramirez, Energy efficiency vs. performance of the numerical so-
745 lution of PDEs: An application study on a low-power ARM-based cluster,
746 Journal of Computational Physics 237 (2013) 132–150.
- 747 [3] A. Morari, A. Tumeo, O. Villa, S. Secchi, M. Valero, Efficient sorting on
748 the Tiler architecture, in: International Symposium on Com-
749 puter Architecture and High Performance Computing (SBAC-PAD), IEEE
750 Computer Society, New York, USA, 2012, pp. 171–178.
- 751 [4] B. D. de Dinechin, P. G. de Massasa, G. Lagera *et al.*, A distributed run-
752 time environment for the Kalray MPPA-256 integrated manycore processor,
753 in: International Conference on Computational Science (ICCS), Elsevier,
754 Barcelona, Spain, 2013, pp. 1654–1663.
- 755 [5] A. Varghese, B. Edwards, G. Mitra, A. P. Rendell, Programming the
756 Adapteva Epiphany 64-core network-on-chip coprocessor, in: International
757 Parallel Distributed Processing Symposium Workshops (IPDPSW), IEEE
758 Computer Society, Phoenix, USA, 2014, pp. 984–992.
- 759 [6] E. Tottoni, B. Behzad, S. Ghike, J. Torrellas, Comparing the power and
760 performance of Intel's SCC to state-of-the-art CPUs and GPUs, in: In-
761 ternational Symposium on Performance Analysis of Systems and Software
762 (ISPASS), IEEE Computer Society, New Brunswick, Canada, 2012, pp.
763 78–87.

- 764 [7] E. Francesquini, M. Castro, , P. H. Penna, F. Dupros, H. C. de Fre-
765 itas, P. O. A. Navaux, J.-F. Méhaut, On the energy efficiency and perfor-
766 mance of irregular application executions on multicore, NUMA and many-
767 core platforms, *Journal of Parallel and Distributed Computing* , In Press.
768 doi:10.1016/j.jpdc.2014.11.002.
- 769 [8] M. Castro, F. Dupros, E. Francesquini, J.-F. Méhaut, P. O. A. Navaux, En-
770 ergy efficient seismic wave propagation simulation on a low-power many-
771 core processor, in: *International Symposium on Computer Architecture*
772 *and High Performance Computing (SBAC-PAD)*, IEEE Computer Society,
773 Paris, France, 2014, pp. 57–64.
- 774 [9] E. Francesquini, A. Goldman, J.-F. Mehaut, A NUMA-Aware Runtime
775 Environment for the Actor Model, in: *International Conference on Parallel*
776 *Processing (ICPP)*, IEEE, Lyon, France, 2013, pp. 250–259.
- 777 [10] L. L. Pilla, C. P. Ribeiro, D. Cordeiro, C. Mei, A. Bhatele, P. O. Navaux,
778 F. Broquedis, J.-F. Mehaut, L. V. Kale, A hierarchical approach for load
779 balancing on parallel multi-core systems, *Proceedings of the 41st Interna-*
780 *tional Conference on Parallel Processing 0 (2012)* 118–127.
- 781 [11] M.J. Rashti *et al.*, Multi-core and network aware MPI topology functions,
782 in: *Proceedings of the 18th European MPI Users’ Group conference on*
783 *Recent advances in the message passing interface, EuroMPI’11, 2011*, pp.
784 50–60.
- 785 [12] G. Mercier, J. Clet-Ortega, Towards an efficient process placement policy
786 for mpi applications in multicore environments, in: *Recent Advances in*
787 *Parallel Virtual Machine and Message Passing Interface*, Springer, 2009,
788 pp. 104–115.
- 789 [13] J. McCaLpin, *Paleoseismology*, 2nd Edition, Vol. 95, Academic press, 2009,
790 Appendix 1.
- 791 [14] J. Lysmer, L. Drake, A finite element method for seismology, *Methods in*
792 *Computational Physics* 11 (1972) 181–216.
- 793 [15] D. Komatitsch, J. P. Vilotte, The spectral-element method: An efficient
794 tool to simulate the seismic response of 2D and 3D geological structures,
795 *Bulletin of the Seismological Society of America* 88 (2) (1998) 368–392.
- 796 [16] M. Dumbser, M. Käser, An arbitrary high-order discontinuous Galerkin
797 method for elastic waves on unstructured meshes II: The three-dimensional
798 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.
- 801 [18] E. H. Saenger, N. Gold, S. A. Shapiro, Modeling the propagation of elastic
802 waves using a modified finite-difference grid, *Wave Motion* 31 (1999) 77–92.
- 803 [19] W. Zhang, Z. Zhang, X. Chen, Three-dimensional elastic wave numerical
804 modeling in the presence of surface topography by a collocated-grid finite-
805 difference method on curvilinear grids, *Geophysical Journal International*
806 190 (1) (2012) 358–378.
- 807 [20] H. Aochi, T. Ulrich, A. Ducellier, F. Dupros, D. Michea, Finite differ-
808 ence simulations of seismic wave propagation for understanding earthquake

- 809 physics and predicting ground motions: Advances and challenges, *Journal*
810 *of Physics: Conference Series* 454 (2013) 1–9.
- 811 [21] P. Aubry, P.-E. Beaucamps *et al.*, Extended cyclostatic dataflow program
812 compilation and execution for an integrated manycore processor, in: *Proce-*
813 *dia Computer Science*, Vol. 18, Elsevier, Barcelona, Spain, 2013, Ch. 2013
814 *International Conference on Computational Science*, pp. 1624–1633.
- 815 [22] K. Datta, S. Kamil, S. Williams, L. Oliker, J. Shalf, K. Yelick, Optimization
816 and performance modeling of stencil computations on modern microproces-
817 sors, *SIAM Review* 51 (1) (2009) 129–159.
- 818 [23] M. Bianco, B. Cumming, A generic strategy for multi-stage stencils, in:
819 *International Conference on Parallel and Distributed Computing (Euro-*
820 *Par)*, ACM, Porto, Portugal, 2014, pp. 584–595.
- 821 [24] G. Rivera, C.-W. Tseng, Tiling optimizations for 3D scientific computa-
822 tions, in: *Supercomputing Conferece (SC)*, ACM, Dallas, USA, 2000, pp.
823 32–38.
- 824 [25] J. Reinders, J. Jeffers, *High Performance Parallelism Pearls: Multicore*
825 *and Many-core Programming Approaches*, 1st Edition, Morgan Kaufmann,
826 2014.
- 827 [26] F. Dupros, C. Pousa, A. Carissimi, J.-F. Méhaut, Parallel simulations of
828 seismic wave propagation on NUMA architectures, in: *International Con-*
829 *ference on Parallel Computing (ParCo)*, IOS Press, Lyon, France, 2010, pp.
830 67–74.
- 831 [27] D. Michéa, D. Komatitsch, Accelerating a three-dimensional finite-
832 difference wave propagation code using GPU graphics cards, *Geophysical*
833 *Journal International* 182 (1) (2010) 389–402.
- 834 [28] M. Horowitz, T. Indermaur, R. Gonzalez, Low-power digital design, in:
835 *Low Power Electronics*, 1994. Digest of Technical Papers., IEEE Sympos-
836 *ium*, 1994, pp. 8–11. doi:10.1109/LPE.1994.573184.
- 837 [29] A. J. Martin, Towards an energy complexity of computation, *Information*
838 *Processing Letters* 77 (2–4) (2001) 181 – 187. doi:http://dx.doi.org/
839 10.1016/S0020-0190(00)00214-3.
- 840 [30] M. Hähnel, B. Döbel, M. Völz, H. Härtig, Measuring energy consump-
841 tion for short code paths using RAPL, *ACM SIGMETRICS Performance*
842 *Evaluation Review* 40 (3) (2012) 13–17.
- 843 [31] G. Lawson, M. Sosonkina, Y. Shen, Energy evaluation for applications with
844 different thread affinities on the Intel Xeon Phi, in: *Workshop on Appli-*
845 *cations for Multi-Core Architectures (WAMCA)*, IEEE Computer Society,
846 2014, pp. 54–59.
- 847 [32] P. Micikevicius, 3D finite-difference computation on GPUs using CUDA, in:
848 *Workshop on General Purpose Processing on Graphics Processing Units*,
849 *ACM*, Washington, USA, 2009, pp. 79–84.
- 850 [33] F. Dupros, H.-T. Do, H. Aochi, On scalability issues of the elastodynamics
851 equations on multicore platforms, in: *International Conference on Compu-*
852 *tational 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.
- 857 [35] M. Christen, O. Schenk, Y. Cui, PATUS for convenient high-performance
858 stencils: evaluation in earthquake simulations, in: Supercomputing Con-
859 ference (SC), IEEE Computer Society, 2012, pp. 11:1–11:10.
- 860 [36] R. Abdelkhalek, H. Calandra, O. Coulaud, G. Latu, J. Roman, Fast seismic
861 modeling and reverse time migration on a graphics processing unit cluster,
862 Concurrency and Computation: Practice and Experience 24 (7) (2012)
863 739–750.
- 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 Conference (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. Leiser-
869 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.
- 872 [39] A. D. Pereira, L. Ramos, L. F. W. Góes, PSkel: A stencil programming
873 framework for CPU-GPU systems, Concurrency and Computation: Prac-
874 tice and Experience.doi : 10.1002/cpe.3479.