# Energy Efficient Seismic Wave Propagation Simulation on a Low-power Manycore Processor.

Marcio Castro, Fabrice Dupros, Emilio Francesquini, Jean-François Mehaut, Philippe Olivier Alexandre Navaux

# Energy Efficient Seismic Wave Propagation Simulation on a Low-power Manycore Processor

Márcio Castro*†, Fabrice Dupros‡, Emilio Francesquini§¶‖, Jean-François Méhaut‖ and Philippe O. A. Navaux*
*Informatics Institute, Federal University of Rio Grande do Sul, Brazil
†Department of Informatics and Statistics, Federal University of Santa Catarina, Brazil
‡BRGM, Orléans, France
§Institute of Computing, University of Campinas, Brazil
¶Institute of Mathematics and Statistics, University of São Paulo, Brazil
‖CEA/DRT, University of Grenoble, France
Email: mbcastro@inf.ufrgs.br, f.dupros@brgm.fr, francesquini@ic.unicamp.br, mehaut@imag.fr, navaux@inf.ufrgs.br

***Abstract*—Large-scale simulation of seismic wave propagation is an active research topic. Its high demand for processing power makes it a good match for High Performance Computing (HPC). Although we have observed a steady increase on the processing capabilities of HPC platforms, their energy efficiency is still lacking behind. In this paper, we analyze the use of a low-power manycore processor, the MPPA-256, for seismic wave propagation simulations. First we look at its peculiar characteristics such as limited amount of on-chip memory and describe the intricate solution we brought forth to deal with this processor's idiosyncrasies. Next, we compare the performance and energy efficiency of seismic wave propagation on MPPA-256 to other commonplace platforms such as general-purpose processors and a GPU. Finally, we wrap up with the conclusion that, even if MPPA-256 presents an increased software development complexity, it can indeed be used as an energy efficient alternative to current HPC platforms, resulting in up to 71% and 81% less energy than a GPU and a general-purpose processor, respectively.**

## I. Introduction

Simulations of large scale seismic wave propagation are very important for risk mitigation, assessment of damage in future hypothetical earthquake scenarios, and oil and gas exploration. Realistic seismic wave propagation simulations rely on complex models, which demand intensive computations on large amounts of data. In this context, High Performance Computing (HPC) appears as the main solution to achieve reliable results in a reasonable amount of time.

Until the last decade, the performance of HPC architectures has been quantified almost exclusively by their processing power, which is usually measured by the number of floating-point operations per second (or Flops). Nowadays, in some contexts, energy efficiency (Flops/Watt) is as important as processing power and has become a critical aspect to the development of scalable systems. On data-centers, for instance, power and cooling costs largely dominate the operational costs (approximately 30% of the energy is used in cooling and 10-15% is lost in power conversions and distribution losses [1]). Taking these aspects into consideration, the official DARPA/IPTO report [2] emphasized that the acceptable power budget to reach the exascale would be 20 MW, which means that an HPC architecture should be able to perform 50 GFlops/W. Yet, the number one ranked machine in the last (November/2013) Green500[1] list (TSUBAME-KFC) has an efficiency of 4.5 GFlops/W, *i.e.*, it is at least ten times less efficient than DARPA/IPTO's recommendation.

The scientific community has been seeking alternatives to lower current power consumption [3], [4]. Recently, a new class of highly-parallel processors called *light-weight manycore processors* was unveiled. Tilera Tile-Gx [5] and Kalray MPPA-256 [6] are examples of such processors, providing high levels of parallelism with hundreds or even thousands of cores. Differently from Graphics Processing Units (GPUs), they feature autonomous cores that can be used to accomplish both data and task parallelism.

Although manycore processors may present better energy efficiency than state-of-the-art general-purpose multicore processors [7], they can also make the development of efficient scientific parallel applications a challenging task due to their architectural idiosyncrasies. 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, efficient execution on these processors requires data transfers to be in conformance with the Network-on-Chip (NoC) topology to mitigate the, otherwise high, communication costs.

In this paper we outline the architectural distinctiveness of the 256-core MPPA-256 processor. Considering these characteristics, we describe how the main kernel of a seismic wave propagation simulator was adapted to 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. Next, taking as a basis optimized GPU and general-purpose processor implementations for this kernel, we show that light-weight manycore processors can be an appealing energy-efficient alternative to seismic wave propagation simulation. Our results show that the solution we propose for

[1]The Green500 (http://www.green500.org) provides a ranking of the most energy-efficient supercomputers in the world.

the MPPA-256 processor can improve the energy efficiency of current solutions using GPUs in 71% and in 81% when compared to general-purpose processors.

The paper proceeds as follows. Section II discusses the fundamentals of seismic wave propagation and its standard approaches. Then, Section III presents the MPPA-256 many-core processor and discusses the challenges we must overcome when dealing with parallel seismic wave propagation on this processor. Section IV presents our approach to perform seismic wave propagation simulations on MPPA-256. Section V discusses the performance and energy efficiency results. Section VI describes related work and Section VII concludes this paper.

## II. Background

In this section we first give a brief introduction to seismic wave propagation. Then, we describe the governing equations and discuss some of their standard parallel implementations.

### A. Seismic Wave Propagation

Predicting strong ground shaking for moderate and large earthquakes is essential for quantitative seismic hazard assessment. Simulations of seismic wave propagations are often constrained by the computational and storage capacity of the hardware platform. Thus seismologists often reduce the scope of the simulation to a small volume instead of considering the whole Earth. The memory footprint of each simulation depends on both the domain of interest and the number of grid points per wavelength. This last condition guarantees numerical stability and therefore the accuracy of the simulation. In our case, we limit our simulations to problem domains that fit into the 2 GB of memory available on the MPPA-256 platform. For that, we perform regional scale modeling spanning a few hundred kilometers in each spatial direction.

In this paper we describe the earthquake process as elastodynamics and we use a finite-differences scheme for solving the wave propagation problem in elastic media [8]. This approach was first proposed in 1970 and since then it has been widely employed due to its simple formulation and implementation.

### B. Governing Equations and Numerical Scheme

The seismic wave equation in the case of an elastic material is:

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

and the constitutive relation in the case of an isotropic medium is:

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

where indices $i$, $j$, $k$ represent a component of a vector or tensor field in cartesian coordinates $(x, y, z)$, $v_i$ and $\sigma_{ij}$ represent the velocity and stress field respectively, and $F_i$ denotes an external source force. $\rho$ is the material density and $\lambda$ and $\mu$ are the elastic coefficients known as Lamé parameters. A time derivative is denoted by $\frac{\partial}{\partial t}$ and a spatial derivative with respect to the $i$-th direction is represented by $\frac{\partial}{\partial i}$. The Kronecker symbol $\delta_{ij}$ is equal to 1 if $i = j$ and zero otherwise.

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

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

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

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

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

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

with $a_1$, $a_2$ and $a_3$ defined as three constants.

The finite difference scheme has been generally developed for equally-spaced structural grids. Thus, the largest numerical dimension of the theoretical problem considered in the rest of this paper is 5.4 million grid points. The computational domain is therefore a cube with 180 grid points in each direction. At this stage, we remind that the analysis presented in this paper is restricted to the parallel performance of the seismic numerical kernel on the MPPA-256 processor. Nevertheless, the results obtained on MPPA-256 processor have been validated using reference optimized implementations for x86 and GPU platforms, as described in the next section.

### C. Parallel implementations

The best parallelization strategy for the elastodynamics equations highly depends on the characteristics of the underlying hardware architecture. In this section, we detail the parallelization strategy used by the reference implementations on multicore and GPUs which are employed throughout the rest of this paper. The reference implementations are those from *Ondes3D*, a seismic wave propagation simulator developed by the French Geological Survey (BRGM).

On shared memory multicore processors, the best way to extract the inherent parallelism of elastodynamics equations is to exploit the triple nested loops coming from the three dimensions of the problem under study. This allows a very straightforward use of OpenMP directives. However, additional optimizations must be employed on large scale Non-Uniform Memory Access (NUMA) architectures, where the memory access time depends on the memory location relative to the processor. In this paper, we rely on classical optimizations for careful data and thread placement using Linux's default first-touch memory policy. We exploit the regular memory access pattern of the finite-differences method to pin threads to cores and to place the data closer to the threads that access it. This guarantees that most memory accesses will be local [9].

However, on GPUs the main difficulty to implement elastodynamics equations comes from the fourth-order stencil. The finite-differences method implies that an important number of accesses to global memory (at least 13) are needed on the GPU to handle each grid point. On the other hand, it is possible to significantly reduce this number if we take into consideration that threads belonging to the same block can access common values using the, faster, shared memory. This drastically improves performance due to a better exploitation of the spatial parallelism. To further improve data reuse, the implementation we employed resorts to a sliding-window algorithm that relies on a two-dimensional tiled decomposition of the three-dimensional domain [10].

## III. The MPPA-256 Manycore Processor

The MPPA-256 is a single-chip manycore processor developed by Kalray that integrates 256 user cores and 32 system cores in 28nm CMOS technology running at 400 MHz. These cores are distributed across 16 compute clusters and 4 I/O subsystems that communicate through data and control Networks-on-Chip (NoCs). This processor has successfully been used in some classes of embedded parallel applications such as signal processing and video decoding [6], [7].

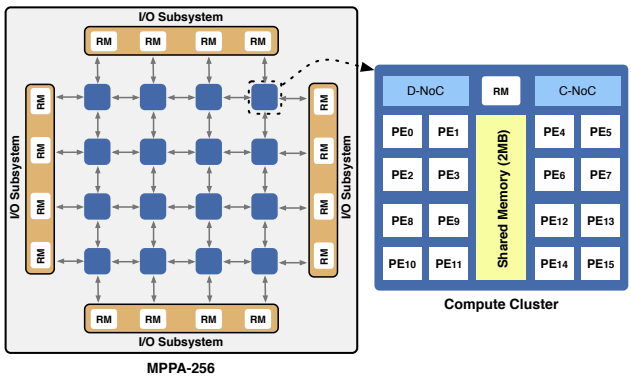



Figure 1: A simplified view of MPPA-256.

Figure 1 shows an architectural overview of the MPPA-256 processor. It features two types of cores: Processing Elements (PE) and Resource Managers (RM). Although RMs and PEs implement the same Very Long Instruction Word (VLIW) architecture, they have different purposes: PEs are dedicated to run user threads (one thread per PE) in non-interruptible and non-preemptible mode whereas RMs execute kernel routines and services of NoC interfaces. Operations executed by RMs vary from task and communication management to I/O data exchanges between either external buses (e.g. PCIe) or DRAM. Both PEs and RMs have private 2-way associative 32 KB instruction and data caches.

PEs and RMs are grouped within compute clusters and I/O subsystems. Each compute cluster has 16 PEs, 1 RM and a low-latency shared memory of 2 MB which enables a high bandwidth and throughput between PEs within the same compute cluster. Each I/O subsystem relies on 4 RMs with a shared D-cache and static memory. The board used in our tests has one of the I/O subsystems connected to an external LP-DDR3 of 2 GB. Contrary to the RMs available on compute clusters, the RMs of I/O subsystems can also run user code.

An important difference of the MPPA-256 architecture is that it does not provide cache coherence between PEs, even among those in the same compute cluster. This means that applications must explicitly maintain the data consistency between PEs private caches. Another difference concerns the Operating System (OS). Each I/O subsystem runs an instance of the Real Time Executive for Multiprocessor System (RTEMS)[2]. It is an open source Real Time OS that supports a variety of open standards such as POSIX and BSD sockets. On the other hand, compute clusters run a proprietary OS called NodeOS. NodeOS implements an Asymmetric Multi-Processing (AMP) architecture, which benefits from the asymmetry that naturally exists in the clusters between the RM and the PEs. Since a different instance of the OS runs in each cluster, about 500 KB of memory is always in use by the OS leaving about 1.5 MB free to the application data and code. Additionally, neither of these OSs support virtual memory and memory transferences from the DDR memory to the local caches must be done explicitly by the application code.

Parallel applications running on MPPA-256 usually follow the *master/worker* model. The *master* process runs on an RM of the I/O subsystem and it is responsible for spawning *worker* processes. Workers are executed on compute clusters and each worker may create up to 16 POSIX threads, one for each PE.

### A. Challenges

The MPPA-256 is an appealing alternative to current general-purpose multicore processors and GPUs, since it features hundreds of cores operating within a very low power budget. However, developing efficient parallel scientific applications for this processor is challenging due to some of its intrinsic characteristics. In this section, we highlight the challenges that must be overcome to efficiently perform parallel seismic wave propagation simulations on this processor.

**Memory management.** MPPA-256 has a limited amount of low-latency memory per compute cluster (2 MB), which is shared by 16 PEs. These memories act as caches, whose goal is to store data coming from the DDR to be computed by PEs. However, data transfers between the DDR and compute clusters' low-latency memories must be explicitly managed by the programmer. Moreover, the data needed for simulating real wave propagation scenarios do not fit into these memories.

[2]http://www.rtems.org

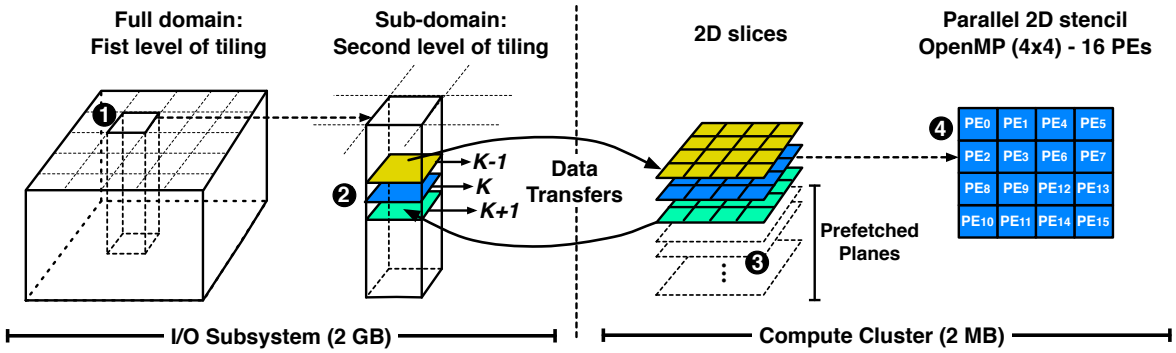



Figure 2: Two-level tiling scheme to exploit the memory hierarchy of MPPA-256.

Thus, the global data must be tiled and transferred between the DDR and the compute clusters' during computation. Next section discusses our approach to deal with this memory constraint.

**Overlapping data transfers.** MPPA-256 allows both synchronous and asynchronous data transfers. A specific API allows the master process (running on the I/O subsystem connected to the DDR) to write data to compute clusters' memory as well as PEs to write the result back to the DDR after computation. Real wave propagation simulations demand several data transfers. Thus, we only employ asynchronous data transfers to alleviate communication costs by overlapping communication with computation.

**Matching NoC topology and communication.** Compute clusters and I/O subsystems are connected by two parallel NoCs, one for data (D-NoC) and another for control (C-NoC). There is one NoC node per compute cluster whereas there are 4 NoC nodes per I/O subsystem. To achieve high bandwidth, the master process running on the I/O subsystem must explicitly select which NoC node to use for the data transfer in accordance to the NoC topology and the application communication pattern. Additionally, it is preferable to perform few data transfers containing large amounts of data instead of several data transfers containing few data to reduce communication costs. These characteristics also guided our algorithmic and implementation decisions discussed in the following section.

## IV. Elastodynamics Numerical Kernel on MPPA-256

Performing stencil computations on the MPPA-256 processor is a challenging task. This class of numerical kernels has an important demand for memory bandwidth. This makes the efficient use of the low-latency memories distributed among compute clusters indispensable. In contrast to standard x86 processors in which it is not uncommon to find last-level cache sizes of tens of megabytes, the MPPA-256 has only 32 MB of low-latency memory divided into 2 MB chunks spread throughout the 16 compute clusters. These chunks of memory are directly exposed to the programmer that must explicitly control them. Indeed, the efficiency of our algorithm relies on the ability to fully exploit this low-latency memories.

On classical distributed memory architectures, the standard parallel implementations of the elastodynamics equations are based on MPI Cartesian grid decomposition. The strategy is based on data-parallelism in which each processor solves its own subdomain problem. The time-dependent computational phase corresponding to the resolution of the first-order system of equations (1) and (2) is the following: at each time step, the stress variables are computed first, then each domain exchanges interface information with its neighbors and finally the velocity variables are updated again by exchanging information on the edges [11].

As we mentioned earlier, the 3D data required for seismic wave modeling do not fit in MPPA-256 low-latency memories making this standard distributed approach ineffectual. Therefore, we need to design efficient master-to-slave and slave-to-master communications to make use of the 2 GB of memory connected to the I/O subsystem and carefully overlap communications with computations to mask communication costs. For that, we implemented a two-level algorithm that decomposes the problem with respect to the memory available on both I/O subsystem and compute clusters. Figure 2 illustrates the general idea of our two-level tiling scheme.

The three dimensional structures corresponding to the velocity and stress fields are allocated on the DDR connected to the I/O subsystem to maximize the overall problem size that can be simulated. Next, we divide the global computational domain into several subdomains corresponding to the number of compute clusters involved in the computation (Figure 2-❶). This decomposition provides a first level of data-parallelism. To respect the width of the stencil (fourth-order), we maintain an overlap of two grid points in each direction. These regions, called ghost zones, are updated at each stage of the computation with point-to-point communications between neighboring clusters. Unfortunately, this first level of decomposition is not enough because three-dimensional tiles do not fit into the 2 MB of low-latency memories available to the compute clusters.

A second level of decomposition is therefore required. We performed this decomposition along the vertical direction as we tile each three-dimensional subdomain into 2D slices (Figure 2-❷). This leads to a significant reduction of the memory consumption for each cluster but requires maintaining a good balance between the computation and communication. Our solution relies on a sliding window algorithm that traverses the 3D domains using 2D planes and overlaps data transfers with computations. This can be viewed as an *explicit prefetching mechanism* (Figure 2-❸) as 2D planes required for the computation at one step are brought to the clusters during the computation performed at previous steps.

Although our solution is flexible enough to handle a variable number of prefetched planes, the maximum value depends on the problem dimensions and the amount of available memory on each compute cluster. To better exploit the NoC, we carefully select the NoC node on the I/O subsystem with which the compute cluster will communicate. This choice is based on the NoC topology and aims at the reduction of the number of hops necessary to deliver a message. Moreover, the prefetching scheme also allows us to send less messages containing more data, which has been empirically proven to be more efficient than sending several messages of smaller sizes due to the important communication latency imposed by the NoC. Finally, OpenMP directives are employed inside each compute cluster to compute 2D problems with up to 16 PEs in parallel (Figure 2-❹).

## V. Experimental Results

In this section, we evaluate the performance and energy consumption of seismic wave propagation simulations on MPPA-256 as well as other common-place platforms. Section V-A presents the measurement methodology and the platforms used in this study. Experimental results are discussed in Sections V-B and V-C. Section V-B concentrates on the performance and energy efficiency aspects of the execution whereas Section V-C discusses the scalability of the proposed algorithm.

### A. Methodology

We analyze the energy efficiency and performance of our seismic wave propagation on MPPA-256 and three other platforms:

**Xeon E5.** This platform features a Xeon E5-4640 Sandy Bridge-EP processor, which has 8 physical cores running at 2.40GHz and 32GB of DDR3 memory.

**Altix UV 2000.** The SGI Altix UV 2000 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 32GB of DDR3 memory shared in a ccNUMA fashion (NUMAlink6). Overall, this platform has 192 CPU cores.

**Quadro K4000.** The NVIDIA Quadro K4000 graphics board features NVIDIA Kepler architecture with 768 CUDA parallel-processing cores running at 800MHz and 3GB of GDDR5 GPU memory.

We use three metrics to compare the energy and computing performance: *time-to-solution*, *energy-to-solution* and *speedup*. Time-to-solution is the time spent to reach a solution for a seismic wave propagation simulation. Analogously, energy-to-solution is the amount of energy spent to reach a solution for a seismic wave propagation simulation. Speedups are calculated by dividing the time-to-solution of the sequential version by time-to-solution of the parallel/distributed version with $n$ cores. All results represent averages of 30 runs to guarantee statistically relevant values.

The energy-to-solution was obtained through each platform's specific power measurement sensors. Both Xeon E5 and Altix UV 2000 are based on Intel's Sandy Bridge microarchitecture. This microarchitecture has Intel's Running Average Power Limit (RAPL) interface which allows us to measure the power consumption of CPU-level components through hardware counters. Power measurements obtained from RAPL are very accurate as shown in [12], [13]. We used this approach to obtain the energy consumption of the whole CPU package including cores and cache memory. Analogously, Quadro K4000 has NVIDIA's Management Library (NVML), an API for monitoring and managing various states of the NVIDIA GPU devices. We used the NVML to gather the power usage for the GPU and its associated circuitry (*e.g.*, internal memory). On Kepler GPUs the reading is accurate to within $\pm 5\%$ of current power draw[3]. Finally, MPPA-256 features a tool called K1-POWER to collect energy measurements of the whole chip, including all clusters, on-chip memory, I/O subsystems and NoCs. We used this tool to measure the energy consumption of MPPA-256 when running the seismic wave propagation kernel. According to the Kalray's reference manuals, the measurement precision on MPPA-256 is $\pm 0.25$ W.

### B. General Energy and Performance Results

In this section we compare the performance and energy consumption of our solution on MPPA-256 against the other multicore and GPU reference implementations discussed in Section II-C. On Xeon E5, we used the solution proposed in [9], which applies OpenMP to parallelize the seismic wave propagation kernel. On Quadro K4000, on the other hand, we used the solution proposed in [10], which is a state-of-the-art parallel solution for seismic wave propagation simulations on GPUs. Figure 3 compares the time-to-solution and energy-to-solution across the processors using a problem size of 2 GB ($180^3$ grid points) and 500 time steps.

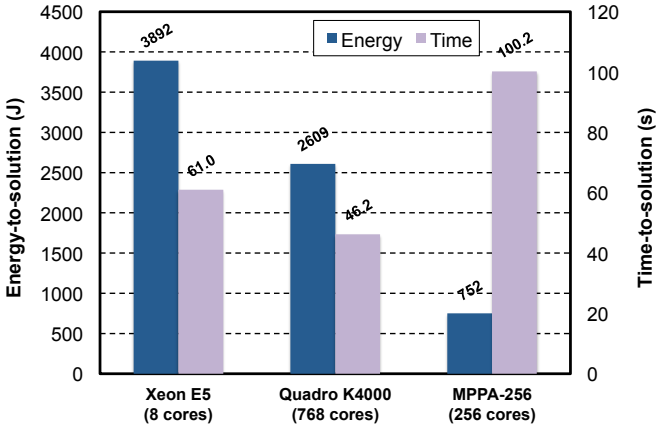



Figure 3: Chip-to-chip comparison.

To the best of our knowledge, GPUs are among the most energy efficient solutions currently in use for seismic wave propagation simulation. However, the important amount of registers and shared memory required by seismic wave kernels on GPUs leads to low overall core occupancy which in turn results in limited performance. Yet, our proposed solution on MPPA-256 achieves the best energy-to-solution among the analyzed processors, consuming 71% less energy than Quadro

[3]NVML reference manual: http://docs.nvidia.com/deploy/pdf/NVML_API_Reference_Guide.pdf

K4000 and 81% less energy than Xeon E5. MPPA-256 is a low-power embedded processor optimized for energy consumption that in theory presents the best Flops/W ratio. Our results corroborate this fact experimentally.

When we consider the time-to-solution, on the other hand, every other processor achieved better performance. One of the reasons for this significant difference in the execution times is the time spent in communications. The simulation algorithm is memory bound and, in our tests, communication accounted for at least 58% of the wall execution time on MPPA-256. Contrary to the other architectures, the small (2 MB) amount of memory available at each compute cluster on MPPA-256 obliges us to perform an important number of data transfers from/to the DDR. Due to the limited on-chip memory, were able to prefetch only up to eight planes before exhausting the available local memory in each compute cluster. Additional factors to the observed performance difference of Xeon E5 and Quadro K4000 in comparison to MPPA-256 include the working frequency of the cores (6x and 2x higher on Xeon E5 and Quadro K4000, respectively) and the number of cores (3x more on Quadro K4000).

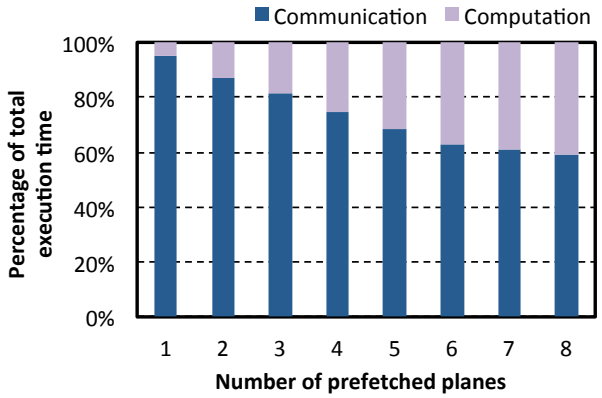



Figure 4: Prefetching impact.

Figure 4 shows the impact of the number of prefetched planes on communication and computation times. As we increase the number planes available at the compute cluster memory level, we improve the reuse of data in the vertical direction. This allows us to overlap a considerable portion of data transfers with computations. However, we observed only slight improvements past six planes. This is due to the saturation of the NoC as this strategy increases the data traffic each time we increase the number of prefetched planes.

### C. Algorithm Scalability Analysis

In the previous section we compared the overall performance and energy consumption of the seismic wave propagation kernel on MPPA-256 against other processors (Xeon E5 and Quadro K4000). In this section we extend the previous analysis by comparing the scalability of our solution on MPPA-256 to the one obtained using an established solution on an HPC platform: the Altix UV 2000. To isolate the communication overhead imposed by the NoC on MPPA-256, we first compare the performance scalability of the parallel version (OpenMP) on a single compute cluster of MPPA-256 against a single node of Altix UV 2000. For this experiment, we used a restricted input 3D space of size $16^3$ grid points to be able to allocate all data in to the compute cluster memory (2 MB) and 50000 time steps. Figure 5(a) compares both the speedup and the energy-to-solution with this problem size on both platforms. To make a fair comparison, we measured the energy consumed by the entire chip on MPPA-256 and only the energy consumed by a single processor of Altix UV 2000.

Figure 5(a) shows that our seismic wave propagation algorithm has similar performance scalability on a single processor of Altix UV 2000 and a single MPPA-256 compute cluster. Considering the energy consumption, we observed that the energy consumption may significantly vary depending on the number of cores/PEs used. When comparing the energy efficiency of a single processor of Altix UV 2000 against a single MPPA-256 compute cluster, we noticed that the former outperformed MPPA-256 on low core counts. For more than 12 cores, however, the MPPA-256 compute cluster consumed up to 17.3% less energy with 16 cores. This comes from the fact that the power consumed by a single processor of Altix UV 2000 almost doubled from 2 to 8 cores (from 18 W to 30 W) whereas the power consumed by the whole MPPA-256 chip increased only slightly from 1 PE to 16 PEs (from 4.1 W to 5 W), while achieving the same scalability.

Figure 5(b) presents the weak scalability of the seismic wave propagation kernel on Altix UV 2000 and MPPA-256 when varying the number of nodes (on Altix UV 2000) and clusters (on MPPA-256). In this case the problem size assigned to each node/cluster stays constant as we increase the number of nodes/clusters. Thus, linear scaling is achieved if the execution time stays constant at 1.00 while the workload is increased in direct proportion to the number of nodes/clusters. As it can be noticed, Altix UV 2000 achieved almost linear scaling. On MPPA-256, however, we observed a considerable performance degradation as we increased the number of clusters. Although the prefetching scheme considerably hides the communication costs on MPPA-256, the latency and bandwidth of the NoC still hurts its performance and thus limits the scalability.

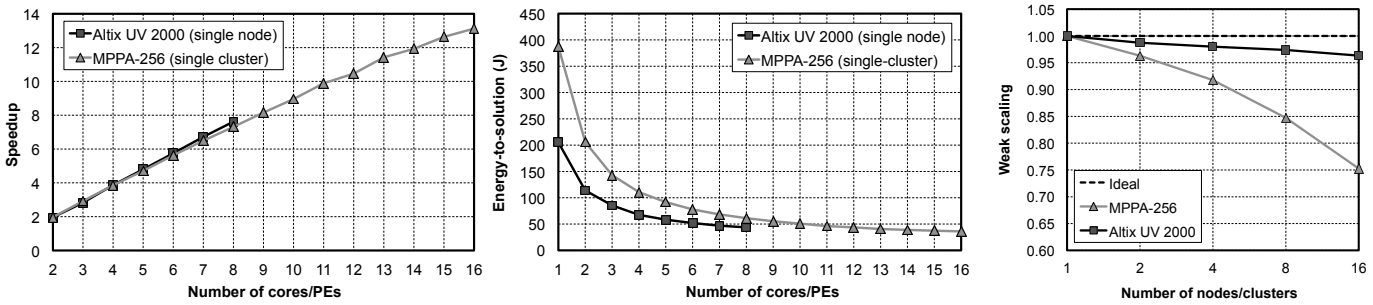
Table I: Time-to-solution and energy-to-solution using a problem size of 2 GB ($180^3$ grid points) and 500 time steps.

| Platform | Time-to-Solution | Energy-to-Solution |
|---|---|---|
| MPPA-256 | 100.2 s | 752 J |
| Altix UV 2000 | 2.9 s | 4418 J |

Table I compares the time-to-solution and energy-to-solution on Altix UV 2000 and MPPA-256 using a problem size of 2 GB ($180^3$ grid points) and 500 time steps. As expected, Altix UV 2000 presented much better performance than MPPA-256, since it has 24 performance optimized general-purpose multicore processors. However, MPPA-256 consumed 83% less energy than Altix UV 2000 to compute the same instance of the input problem.

## VI. Related Work

**Low-power manycore processors.** The use of low-power manycore processors for HPC is a topic of ongoing research. *Totoni et al.* [14] compared the power and performance of Intel's Single-Chip Cloud Computer (SCC) to other types of




(a) Altix UV 2000 single-processor vs. MPPA-256 single-cluster.

(b) Scalability on both platforms when varying the number of nodes/clusters.

Figure 5: Scalability and energy consumption: Altix UV 2000 vs. MPPA-256.

CPUs and GPUs. The analysis was based on a set of parallel applications implemented with the Charm++ programming model. Although they showed that there is no single solution that always achieves the best trade-off between power and performance, the results suggest that manycores are an opportunity for the future. *Morari et al.* [5] proposed an optimized implementation of radix sort for the Tilera TILEPro64 manycore processor. The results showed that the their solution for TILEPro64 provides much better energy efficiency than an general-purpose multicore processor (Intel Xeon W5590) and comparable energy efficiency with respect to a GPU NVIDIA Tesla C2070. *Castro et al.* [15] showed that MPPA-256 can be very competitive considering both performance and energy efficiency for a fairly parallelizible problem: the Traveling-Salesman Problem (TSP). While this problem is CPU-bound, it also displays important issues related to imbalance and irregularity. The results showed MPPA-256's performance to be roughly equivalent to that of an Intel Xeon E5-4640 processor, which has 8 CPU cores (16 threads with Hyper-Threading) running at 2.40GHz while consuming approximately 13 times less energy. Using a different approach, *Aubry et al.* [7] compared the performance of an Intel Core i7-3820 processor against MPPA-256. Their application, an H.264 video encoder, was implemented using a dataflow language that offers direct automatic mapping to MPPA-256. Their findings show that the performance of these traditional processors is on par with the performance of the MPPA-256 embedded processor which provides 6.4x better energy efficiency.

In contrast to those previous works, we intended to assess how well this processor fares when facing a memory-bound application. We focused on the challenges and on the algorithmic aspects of the use of the MPPA-256 processor in the context of energy efficient scientific computing. Additionally, we described some of the programming issues that must be considered when developing parallel applications to MPPA-256.

**Seismic wave propagation.** Adapting seismic wave propagation numerical kernels to emerging parallel architectures is an active research topic. Due the versatility of the finite-differences method used both for oil and gas applications and standard earthquakes modeling, several strategies have been proposed. For instance, Intel and NVIDIA have proposed technical reports devoted to the efficient implementation of such stencils on multicore processors [16], Intel Xeon Phi [17] and GPUs [18]. These contributions underline the impact of low-level optimizations but also introduce efficient spatial blocking for the finite-differences numerical method.

Dupros *et al.* [19] presented a review of the scalability issues for distributed and shared-memory platforms with a focus on mapping processes/threads on hierarchical clusters. Dursun *et al.* [20] introduced several additional strategies, including inter-node and intra-node optimizations. Recently, Christen*et al.* [21] described the use of the Patus framework to optimize the *AWP-ODC* finite difference code. They underlined the impact of vectorization and cache prefetching and reported a performance improvement up to 15% when running the complete application with 512 MPI processes. Several research efforts have been done on the adaptation of seismic wave kernels on GPUs to overcome the poor arithmetic intensity (FLOPS/transferred byte ratio) offered by elastodynamics stencils [10], [22]. The results obtained demonstrate that GPUs could be a relevant alternative to standard general-purpose processors.

In the context of energy efficiency of seismic wave propagation simulations, Krueger *et al.* [23] compared the performance and energy efficiency of an Intel Nehalem platform, an NVIDIA Tesla GPU and a simulated general-purpose manycore chip design optimized for high-order wave equations called "Green Wave". They showed that Green Wave can be up to 8x and 3.5x more energy efficient per node when compared with the Nehalem and GPU platforms respectively. Differently from Krueger *et al.*, our experiments and measurements were carried out on a real manycore processor. MPPA-256 is not optimized for high-order wave equations as Green Wave, however it still achieved important energy efficiency improvements of $\sim$5x and $\sim$71% when compared to a multicore Intel Xeon and NVIDIA Tesla GPU, respectively. Godekke *et al.* [24] compared the energy efficiency and performance of an ARM Cortex-A9 based cluster with a standard x86 cluster to solve three different classes of numerical solution methods for partial differential equations. Although they demonstrated good weak and strong scaling results, they concluded that the energy efficiency of the ARM-based cluster strongly depends on the configuration of the problem under study.

## VII. Conclusion and Perspectives

Simulation of seismic wave propagation is a field of research that demands vast amounts of processing power typically provided by the newest general-purpose CPUs and GPUs. The recent introduction of light-weight manycores, such as

MPPA-256, presents an opportunity to perform highly parallel energy-efficient computations. In this paper, we presented our approach to the simulation of this physical phenomenon using the MPPA-256 processor. MPPA-256 is an embedded low-power manycore processor that has several architectural peculiarities which must be tamed in order to obtain good performance. Due to the limited size of the local memories, we developed a new multi-level tiling strategy and a prefetching mechanism. Their goal is two-fold. Firstly, they allow us to work with data bigger than the local-memory sizes. Secondly, this mechanism lightens the communication overhead imposed by the NoC. We show that our solution can achieve better energy efficiency than currently largely used solutions based on GPUs (improvement of 71%) and general-purpose multicore processors (average improvement of 82%).

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

## ACKNOWLEDGEMENTS

This work was supported by CAPES, the HPC-GA project funded by the FP7-PEOPLE under grant agreement number 295217 and BRGM Carnot-institute.

## REFERENCES

[1] E. Pakbaznia and M. Pedram, "Minimizing data center cooling and server power costs," in *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*. San Fancisco, USA: ACM, 2009, pp. 145–150.

[2] Kogge, Peter *et. al*, "Exascale computing study: Technology challenges in achieving exascale systems," Defense Advanced Research Projects Agency (DARPA), Notre Dame, USA, Tech. Rep., 2008.

[3] N. Rajovic *et. al*, "The low-power architecture approach towards exascale computing," in *Workshop on Scalable Algorithms for Large-Scale Systems (ScalA)*. Seattle, USA: ACM, 2011, pp. 1–2.

[4] D. Göddeke and Dimitri Komatitsch *et al.*, "Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster," *J. Comput. Physics*, vol. 237, pp. 132–150, 2013.

[5] A. Morari, A. Tumeo, O. Villa, S. Secchi, and M. Valero, "Efficient sorting on the Tilera manycore architecture," in *IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. New York, USA: IEEE, 2012, pp. 171–178.

[6] B. D. de Dinechin, P. G. de Massasa, and G. Lagera *et. al*, "A distributed run-time environment for the Kalray MPPA-256 integrated manycore processor," in *International Conference on Computational Science (ICCS)*. Barcelona, Spain: Elsevier, 2013, pp. 1654–1663.

[7] P. Aubry, P.-E. Beaucamps, and F. Blanc *et. al*, "Extended cyclostatic dataflow program compilation and execution for an integrated manycore processor," in *International Conference on Computational Science (ICCS)*. Barcelona, Spain: Elsevier, 2013, pp. 1624–1633.

[8] P. Moczo *et al.*, "The finite-difference time-domain method for modeling of seismic wave propagation," in *Advances in Wave Propagation in Heterogeneous Media*, ser. Advances in Geophysics. Elsevier, 2007, vol. 48, ch. 8, pp. 421–516.

[9] F. Dupros, C. Pousa, A. Carissimi, and J.-F. Méhaut, "Parallel simulations of seismic wave propagation on NUMA architectures," in *International Conference on Parallel Computing (ParCo)*. Lyon, France: IOS Press, 2010, pp. 67–74.

[10] D. Michéa and D. Komatitsch, "Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards," *Geophysical Journal International*, vol. 182, no. 1, pp. 389–402, 2010.

[11] F. Dupros, H. Aochi, A. Ducellier, D. Komatitsch, and J. Roman, "Exploiting intensive multithreading for the efficient simulation of 3D seismic wave propagation," in *International Conference on Computational Science and Engineering (CSE)*. São Paulo, Brazil: IEEE, 2008, pp. 253–260.

[12] E. Rotem *et al.*, "Power-management architecture of the Intel microarchitecture code-named Sandy Bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012.

[13] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using RAPL," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, pp. 13–17, 2012.

[14] E. Totoni and B. Behzad *et. al*, "Comparing the power and performance of Intel's SCC to state-of-the-art CPUs and GPUs," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. New Brunswick, Canada: IEEE, 2012, pp. 78–87.

[15] M. Castro, E. Francesquini, T. M. Nguélé, and J.-F. Méhaut, "Analysis of computing and energy performance of multicore, NUMA, and manycore platforms for an irregular application," in *Workshop on Irregular Applications: Architectures & Algorithms (IA^3)*. Denver, EUA: ACM, 2013, p. Article No. 5.

[16] L. Borges and P. Thierry, "3D Finite Differences on Multi-core Processors," INTEL, Technical Report, 2011.

[17] Leonardo Borges and Philippe Thierry, "Optimize seismic image processing on Intel many integrated core architecture (Intel MIC architecture)," Intel, Technical Report, 2011.

[18] P. Micikevicius, "3D finite-difference computation on GPUs using CUDA," in *Workshop on General Purpose Processing on Graphics Processing Units*. Washington, USA: ACM, 2009, pp. 79–84.

[19] F. Dupros, H.-T. Do, and 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.

[20] H. Dursun, K.-I. Nomura, L. Peng, R. Seymour, W. Wang, R. K. Kalia, A. Nakano, and P. Vashishta, "A multilevel parallelization framework for high-order stencil computations," in *European Conference on Parallel Processing (Euro-Par)*, Delft, Netherlands, 2009, pp. 642–653.

[21] M. Christen, O. Schenk, and Y. Cui, "Patus for convenient high-performance stencils: evaluation in earthquake simulations," in *International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE, 2012, p. Article No. 11.

[22] R. Abdelkhalek, H. Calendra, O. Coulaud, J. Roman, and G. Latu, "Fast seismic modeling and reverse time migration on a GPU cluster," in *International Conference on High Performance Computing & Simulation (HPCS)*. Leipzig, Germany: IEEE, 2009, pp. 36–43.

[23] J. Krueger, D. Donofrio, J. Shalf, M. Mohiyuddin, S. Williams, L. Oliker, and F.-J. Pfreund, "Hardware/software co-design for energy-efficient seismic modeling," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2011, p. Article No. 73.

[24] D. Göddeke, D. Komatitsch, M. Geveler, D. Ribbrock, N. Rajovic, N. Puzovic, and 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*, vol. 237, pp. 132–150, 2013.