Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Towards Porting a Real-World Seismological Application to the Intel MIC Architecture

V. Weinberg[*], M. Allalen and G. Brietzke

Leibniz Rechenzentrum der Bayerischen Akademie der Wissenschaften, 85748 Garching b. München, Germany

Abstract

This whitepaper aims to discuss first experiences with porting an MPI-based real-world geophysical application to the new Intel Many Integrated Core (MIC) architecture. The selected code SeisSol is an application written in Fortran that can be used to simulate earthquake rupture and radiating seismic wave propagation in complex 3-D heterogeneous materials. The PRACE prototype cluster EURORA at CINECA, Italy, was accessed to analyse the MPI-performance of SeisSol on Intel Xeon Phi on both single- and multi-coprocessor level. The whitepaper presents detailed scaling results on EURORA and compares them with the SandyBridge-based HPC system SuperMUC at LRZ at Garching near Munich, Germany. The work was done in a PRACE Preparatory Access project within the PRACE-1IP extension.

1. Introduction

Accelerator-based systems have entered the world of HPC computing since around five years. However, programming accelerators like GPGPUs using OpenCL or CUDA is very cumbersome and error-prone. Trying to overcome these difficulties, Intel developed their own Many Integrated Core (MIC) architecture which can be programmed using standard parallel programming techniques like OpenMP and MPI.
LRZ has been considered by Intel as a leading research centre for evaluating coprocessors based on Intel's new MIC architecture since 2010 under strict NDA. During the evaluation in the past, we mainly concentrated on using OpenMP and Intel libraries like MKL in native and offload mode on single Knights Ferry and Knights Corner prototypes [1]. In the beginning of 2013, Intel Xeon Phi became the first product available on the market. 14 (non-European) systems based on Intel Xeon Phi have entered the Nov. 2013 edition of the TOP500 list [2] headed by the Tianhe-2 system at the National Super Computer Centre in Guangzhou. At the European level an Intel Xeon Phi based prototype cluster EURORA was installed in mid-2013 at CINECA [3] and could be accessed through PRACE Preparatory Access calls. This gave us the possibility to evaluate an Intel Xeon Phi-based cluster in multi-user mode with batch-system integration for the very first time.
The main purpose of our project was to explore the performance and scaling behaviour of a real-world MPI-based application on the new Intel MIC architecture and last but not least test the maturity of Intel Xeon Phi based clusters on the multi-coprocessor level from a computer-centre's perspective. Rather than rewriting critical parts of the code (which was not possible within the very limited timeframe of the project) we tried to improve the performance using compiler options and compiler directives for time-critical code parts, task/thread pinning techniques and hybrid programming models.
The geophysical application SeisSol [4] was selected as a showcase for a real-world MPI-based application. In the past LRZ was involved in improving the load balancing of this code [5] [6] and has given support for the code within the KONWIHR-II project "GeoPF: Geophysics for PetaFlop Computing" [7]. Currently, within the European VERCE project [8] LRZ is involved in improving the quality of seismic application software. Within PRACE-3IP LRZ is leading the subtask to create a Best Practice Guide for Intel Xeon Phi [9].

[*] Corresponding author. *E-mail address*: weinberg@lrz.de

2. Description of the code SeisSol

SeisSol is a research code that can be used to simulate dynamic earthquake sources and emanating seismic waves in complex heterogeneous materials in 3D ([10][11], and references therein). The underlying numerical scheme is based on the ADER-DG approach: the method combines the Discontinuous Galerkin (DG) Finite Element method with the ADER approach using Arbitrary high order DERivatives for flux calculation [12].
The code is written in Fortran and uses MPI for parallelisation. Both, a 2D and a 3D version of the software exist. Here we consider only the 3D version due to its relevance for HPC. The 3D version continues to be a research code under active development. Recently it has been demonstrated that the code has remarkable numerical properties and advantages in comparison to implementations based on other numerical approaches [10][13]. As such it can be used to compare historic with synthetic earthquakes and trace and identify physical model parameters with actual observed phenomena, e.g. the 1992 M7 Landers earthquake in California (see Fig. 1 for a visualisation of large-scale dynamic earthquake scenario simulation on SuperMUC). The results can be used for analyses of the geophysical subsurface structure and geometry, and/or the parameterisation of the implemented constitutive laws and source parameters.

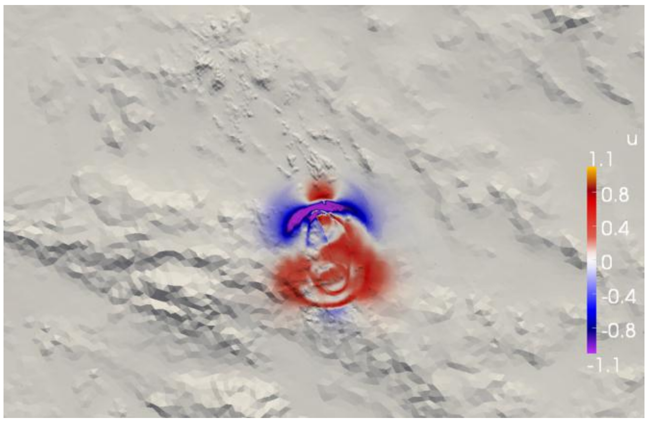


Figure 1: Simulation of the 1992 M7 Landers earthquake in California. The figure shows the velocity along east-west.

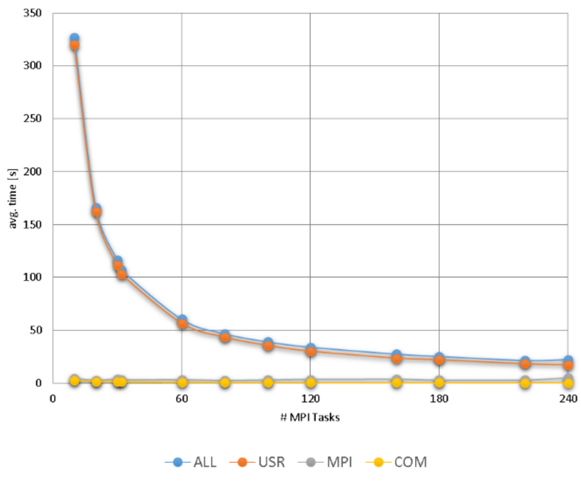
The most commonly used parallelisation strategy for partial differential equations is based on the partitioning of the physical domain in space and therefore the partitioning of the computational mesh. This is often done automatically with graph partitioning methods (like in the METIS software package) and/or with space-filling curves (e.g. Hilbert curves). The sublattices are then handled by the cores/threads on the nodes of a parallel machine. The communication between the cores is implemented using MPI.

Our attempts to simulate a typical realistic use-case (earthquake scenario with dynamic source parametrising a historic earthquake, e.g. the 1992 M7 Landers earthquake in California) failed to run on the Intel Xeon Phi due to the limited memory size. Therefore we simplified and downscaled the setup to a technical valid but physical irrelevant test case using small-scale meshes created by a mesh generator that comes with the software. The generator creates a unit cube with $5 \times n^3$ tetrahedra. Our generated meshes of 5000 (n=10) to 135000 (n=30) elements finally fit the rather small memory size of 8 GB on 1 MIC coprocessor. We have created meshes with size $n \times n \times n$, for n = 10, 15, 20, 25 and 30. Runs with larger n crashed on Intel Xeon Phi. As a further pre-processing step for MPI-parallel runs the meshes were partitioned using the METIS partitioning software for all task sizes in the range 2, …, 240.

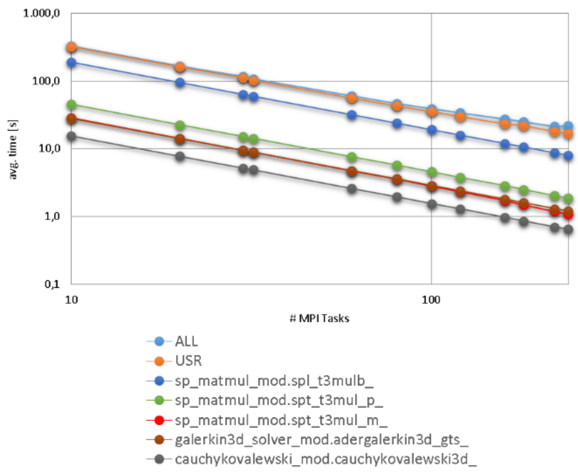
The Scalasca utility [14] has been used to analyse the runtime behaviour of the code. The Scalasca analysis has been performed on the thin node islands of the SuperMUC system at LRZ. Each thin node island consists of 512 SandyBridge-EP Intel Xeon nodes, and each node is equipped with 16 cores clocked with max. 2.7 GHz. The peak performance on SuperMUC is 345.6 GFlops/node.

In Fig. 2 we present scaling results for the n=20 test case on up to 240 cores on SuperMUC. Scalasca was used to break the total measured wall-clock time down into the time spent within MPI (“MPI”), pure user functions (“USR”) and functions calling subprograms or MPI (“COM”). As seen in Subfig. (a) the wall-clock time is completely dominated by “USR”-functions, i.e. functions that do not call other functions or MPI routines. The scaling behaviour of the code on SuperMUC is very good, due to the near linear scaling of the “USR” functions.

Subfig. (b) shows the scaling of the dominating "USR" functions. The main performance bottleneck is the multiplications of relatively small sparse matrices (routines *sp_matmul_mod.**), which dominate the runtime execution.



(a)



(b)

Figure 2: Strong scaling of SeisSol on SuperMUC for the n=20 test case. Subfig. (a) shows the complete time ("ALL"), the portion spent in the MPI routines, the "USR"-functions not calling any subroutines and the "COM"-functions which call subroutines or MPI calls, however the time spent within the "USR" or "MPI" regions is not counted by Scalasca. Subfig. (b) shows the time spent in the dominating "USR"-functions separately.

3. The PRACE prototype EURORA at CINECA

Performance results have been obtained on the PRACE prototype EURORA (EURopean many integrated cORe Architecture) cluster at CINECA, Italy. The current configuration of EURORA consists of 64 compute nodes, each with two 8-core Intel Xeon SandyBridge processors (2.10 and 3.10 GHz) and has either 16 GB (thin nodes) or 32 GB memory (fat nodes). Since each node is equipped with either two Intel Xeon Phi (MIC) cards or two nVIDIA Tesla K20 (Kepler) graphics accelerators, the system in current production provides a total of 1024 SandyBridge cores with 64 Xeon Phi cards and 64 NVIDIA GPUs, and a total of 1.1 TB of memory. For the interconnects between nodes EURORA uses a custom technology based on an FPGA 3D Torus and an Infiniband network. The user environment employs a Linux (RedHat CentOS release 6.3) operating system. The software stack for MIC includes Intel MPSS (only installed on compute nodes), Intel and (unoptimised) GNU compilers, Intel MPI and MKL. User jobs are managed by the PBS Pro batch system which is one of the few batch systems that support Intel MIC officially. During the PRACE-1IP extension period (Jul-Dec 2013) the cluster was open for Preparatory Access for PRACE partners and Task T7.1 "Applications Enabling for Capability Science" in Work Package 7 (WP7) focused on the application enabling support for the capability science projects on the Xeon Phi coprocessors [15].
Details about the installed Intel Xeon Phi coprocessors (B1 stepping) are summarised in the following Table 1:

Number of cores	60
Frequency of cores	1.1 GHz
GDDR5 memory size	8 GB
Number of hardware threads per core	4
SIMD vector registers	32 (512-bit wide) per thread context
Flops/cycle	16 (DP), 32 (SP)
Theoretical peak performance	1 TFlop/s (DP), 2 TFlop/s (SP)
L2 cache per core	512 kB

Table 1: Hardware parameters of the Intel Xeon Phi coprocessors installed in the EURORA cluster at CINECA.

MPI programs for Intel MIC must be compiled with the compiler switch "*-mmic*" to generate code for the MIC architecture. Since the MPSS stack is only installed on the compute nodes, also for compilation an interactive job had to be started, which was suboptimal. CINECA supported both native and host-centric offload mode on their nodes.

In the default host-centric mode MPI programs can be launched from the host by simply using *mpirun.mic* (which is the default *mpirun* coming with Intel MPI). When running on multiple nodes the hostfile of the PBS scheduler (referenced by the variable *$PBS_NODEFILE*) has to be parsed to obtain the names of the reserved hosts. Using a common naming scheme, the hostnames of the Intel Xeon Phi coprocessors can then be derived. To run MPI jobs on e.g. 4 cards using 60 MPI tasks per coprocessor the following steps are necessary

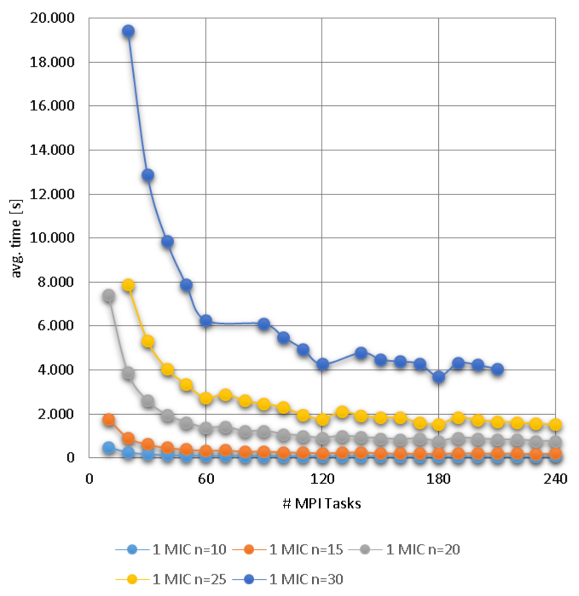
1. *# module load intel intelmpi*
2. *# source $INTEL_HOME/bin/compilervars.sh intel64*
3. *# export I_MPI_MIC=enable*
4. *# export I_MPI_DAPL_PROVIDER_LIST=ofa-v2-mlx4_0-1,ofa-v2-scif0,ofa-v2-mlx4_0-1*
5. *# cat $PBS_NODEFILE*
   node019.eurora.cineca.it
   node023.eurora.cineca.it
6. *# mpirun.mic -host node019-mic0 -n 60 ./seissolxx-mic PARAMETERS.par : -host node019-mic1 -n 60 ./seissolxx-mic PARAMETERS.par : -host node023-mic0 -n 60 ./seissolxx-mic PARAMETERS.par : -host node023-mic1 -n 60 ./seissolxx-mic PARAMETERS.par*

This loads the Intel compiler and Intel MPI environment (step 1), sets Intel specific variables (step 2), enables MPI execution on MIC coprocessors (step 3), sets fabric specific settings (step 4), gets the names of the allocated host nodes (step 5) and finally executes the MPI program on the MIC cards attached to the host nodes node019 and node023. Without the fabrics settings in step 4 suggested by CINECA support all jobs using more than the 2 coprocessors attached to a single host failed.

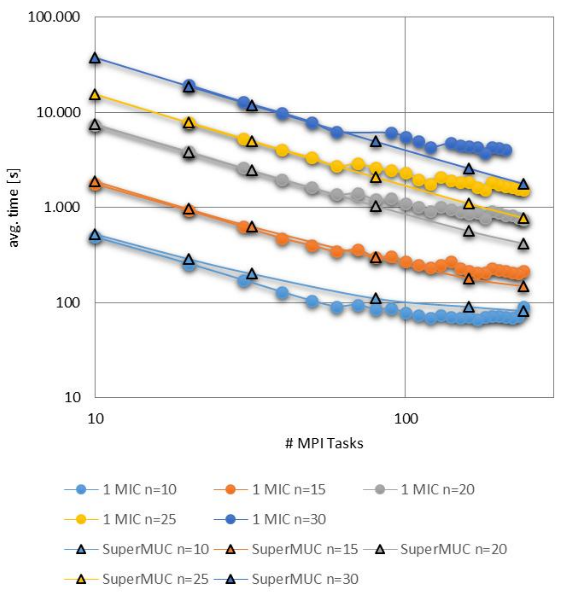
Since the home filesystem and the filesystem containing Intel compiler and MPI libraries are mounted via NFS on the coprocessor, neither the executable nor the input data or libraries need to be copied to the coprocessor. In native mode the MPI program must be launched from a shell running on the coprocessor using *mpiexec.hydra*. In both modes the MPI program is executed on Intel Xeon Phi under the same user ID as on the host. For this purpose on EURORA the */etc/passwd* file on the MIC contains one line for the specific user for whom the coprocessor has been reserved by the batch system.

4. Scaling results

Scaling test-runs have been performed using up to 4 coprocessors with 2 to 240 MPI tasks per coprocessor. The results are compared with scaling results on SuperMUC. Figure 3 shows the timing results on 1 MIC coprocessor (a) and compares them to the scaling on the SandyBridge-EP based system SuperMUC at LRZ (b). Mind that the theoretical peak performance per physical core on SandyBridge-EP (16 cores/node @ 2.7 GHz, 8 DP Flops/cycle) is 21.6 GFlops and on MIC (60 cores/coprocessor @ 1.03 GHz, 16 DP Flops/cycle) is 16.4 GFlops. On up to 60 MIC cores the scaling behaviour is almost identical, in the regime of hardware hyperthreading on MIC (60 < tasks <= 240) linear scaling breaks, but some additional gain in performance is observed especially for larger meshes. However, the overall “out-of-the-box” performance on the MIC coprocessor is a factor of 25 slower than on SandyBridge-EP. Reasons for this factor might be the differences in core frequency, memory and I/O latencies, cache sizes, out-of-order vs. in-order architecture etc.



(a)



(b)

Figure 3: Timing of SeisSol for 5 different mesh sizes. Subfig. (a) shows the execution time on a single MIC coprocessor in dependence of the number of MPI tasks. Subfig. (b) compares the measurements on MIC with the scaling behaviour on SuperMUC. Mind that the timing results on SuperMUC have been multiplied by a factor of 25 to fit in the same figure.

Selected data to deploy these scalability curves is shown in the following Tables 2 (n=10) and 3 (n=20):

Number of MPI tasks/MIC	Wall-clock time [s]	Speed-up vs the first one	Number of MICs	Total number of MPI tasks
10	489.93	1	1	10
20	257.21	1.90	1	20
30	173.41	2.82	1	30
60	91.22	5.37	1	60
120	68.87	7.11	1	120
180	70.76	6.92	1	180
240	89.73	5.46	1	240

Table 2: Measurements on 1 MIC coprocessor (Fig. 3, n=10).

Number of MPI tasks/MIC	Wall-clock time [s]	Speed-up vs the first one	Number of MICs	Total number of MPI tasks
10	7449.47	1	1	10
20	3904.67	1.91	1	20
30	2616.28	2.85	1	30
60	1384.50	5.38	1	60
120	908.98	8.20	1	120
180	778.79	9.57	1	180
240	751.02	9.92	1	240

Table 3: Measurements on 1 MIC coprocessor (Fig. 3, n=20).

Figure 4 (a) shows the preliminary timing measurements on multiple MIC coprocessors vs. the number of the MPI tasks per coprocessor. Using more than one coprocessor did not improve the performance (needs further investigation). Negative scaling was observed when using more than 60 tasks per coprocessor in the multi-coprocessor case.

Scaling numbers for the multi-coprocessor case are shown in the following Table 4:

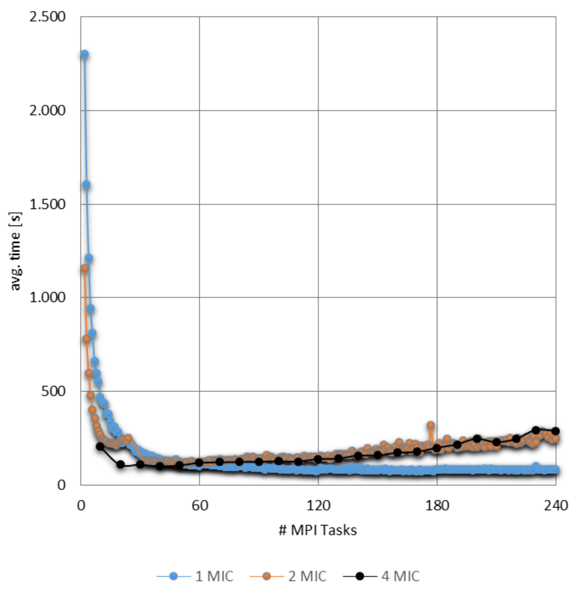
Number of MPI tasks/MIC	Wall-clock time [s]	Speed-up vs the first one	Number of MICs	Total number of MPI tasks
60	101.46	1	1	60
60	120.14	0.84	2	120
60	118.40	0.86	4	240

Table 4: Measurements on multiple MIC coprocessors using 60 tasks per coprocessor (Fig. 4 (a), n=10).

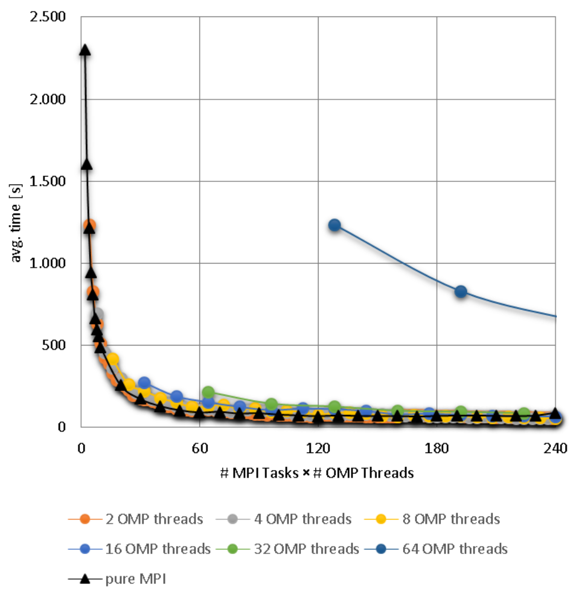
We have also tested an experimental unimproved hybrid MPI/OpenMP version coming with the code[†]. Performance results are shown in Figure 4 (b) and are compared with the pure MPI version of the code. The hybrid version shows lower performance than the MPI-only version, with the performance getting worse with increasing number of OpenMP threads. One possible explanation could be the suboptimal pinning of the threads on the MIC and data partitioning in the job which does not minimize the total effective inter-processor communication.

Using various pinning settings like *I_MPI_PIN_DOMAIN=core* or *node* or executing the code in native mode using *mpiexec.hydra* directly on *tmpfs* on the MIC coprocessor did not change the performance significantly, since the test cases are not dominated by I/O over the PCIe bus.

Using auto-vectorisation and simple introduction of supporting compiler directives did not improve the overall code-performance. The failure is due to the fact that the costly sparse-matrix multiplication compute kernels are implemented using indirect vector-indexing for generality of formulation. Small improvements in performance may be possible when allowing additional code-rewrite and usage of compiler directives within the critical routines. However, in order to achieve significant improvements in performance through vectorisation it is likely necessary to explicitly exploit the different sparsity patterns requiring significant code rewrite.



(a)



(b)

Figure 4: Subfig. (a) shows the timing of SeisSol on multiple MIC coprocessors for the smallest mesh size as a function of the MPI tasks per coprocessor. Subfig. (b) displays the timing of a hybrid version of SeisSol using MPI and OpenMP as a function of the product of the number of MPI tasks and the number of OpenMP threads in comparison with the pure MPI version of the code.

[†] An improved hybrid MPI/OpenMP version is currently under development at TUM, Munich and shows very promising results on SandyBridge.

5. Summary and Outlook

Concerning the usability and the programmability Intel MIC is a promising architecture for scientific computing compared to other accelerator based systems like GPGPUs, FPGAs or former CELL processors or ClearSpeed cards. The possibility to log on to the coprocessor for diagnostic inspection (e.g. to watch the core and memory usage interactively with tools like *top*) was a great help compared to GPGPU based systems. It was a very good experience to work on a MIC-based cluster in multi-user mode with batch-system integration for the very first time. The insight into the configuration was very valuable for the Super-MIC installation at LRZ scheduled in Q1 2014. Regarding the maturity of MIC-clusters, we would consider Intel Xeon Phi based clusters still as prototype systems and not really suited for production runs of real-world MPI based applications yet.
The MPI based version of SeisSol as well as the experimental hybrid MPI/OpenMP version could be quickly ported to the new architecture. Scaling of the MPI-based application SeisSol on up to 60 MPI tasks on one single coprocessor was almost linear and similar to the scaling on SuperMUC, however the performance was slower by a factor of 25. The small GDDR memory size restricted the test cases to rather small (unphysical) meshes and input parameter sets. Attempts to run real-world seismic simulation like the Landers earthquake failed. Likely causes for the suboptimal performance are the failing of efficient auto-vectorisation techniques inside the sparse matrix-matrix multiplication routines due to indirect vector-indexing and cache-size limitations. Therefore considerable efforts would need to be made to exploit the 512-bit vector-units of the MIC-architecture. Native execution on 1 MIC and preliminary scaling tests on multiple coprocessors did not show improvement. Alternative implementations of the compute-kernels by generating vector code using vector-intrinsics are already work in progress and have shown very good scaling on SuperMUC [16]. However, we hope that the next product of the MIC family, announced as “Knights Landing” by Intel, with integrated on-package memory and also functioning in standalone CPU mode, will deliver better “out-of-the-box” performance together with future compiler releases.

References

[1] V. Weinberg, M. Allalen, First experiences with the Intel MIC architecture at LRZ, InSiDE, Vol. 11 No. 2 Autumn 2013, 88-91, *http://inside.hlrs.de/pdfs/inSiDE_autumn2013.pdf*

[2] *http://www.top500.org*

[3] *http://www.hpc.cineca.it/hardware/eurora*

[4] *http://www.geophysik.uni-muenchen.de/~kaeser/SeisSol*

[5] M. Felder, O. Rivera, M. Käser, M. Dumbser, Optimisation of a Novel Seismological Solver Code, LRZ Report 2008-03, *http://www.lrz.de/wir/berichte/TB/LRZ-Bericht-2008-02.pdf*

[6] O. Rivera M. Käser (2008), Towards Optimal Load Balance in Parallel Programs for Geophysical Simulations, InSiDE, Vol. 6 No. 1 Spring 2008, 46-49. *http://inside.hlrs.de/pdfs/inSiDE_spring2008.pdf*

[7] KONWIHR, The Bavarian Competence Network for Technical and Scientific High Performance Computing, *http://www.konwihr.uni-erlangen.de*

[8] VERCE (Virtual Earthquake and seismology Research Community in Europe), *http://www.verce.eu*

[9] V. Weinberg (editor) et al., Best Practice Guide – Intel Xeon Phi, *http://www.prace-ri.eu/IMG/pdf/best-practice-guide-intel-xeon-phi.pdf*

[10] J. de la Puente, J.-P. Ampuero, M. Käser, Dynamic Rupture Modeling on Unstructured Meshes Using a Discontinuous Galerkin Method, J. Geophys. Res., 114, B10302

[11] H. Igel, M. Käser, M. Stupazzini, Simulation of Seismic Wave Propagation in Media with Complex Geometries, in Encyclopedia of Complexity and System Science, edited by WHK Lee, Springer Verlag, 2009

[12] M. Dumbser, M. Käser, An Arbitrary High Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes II: The Three-Dimensional Isotropic Case, Geophysical Journal International, 167(1), 319-336

[13] C. Pelties, J. de la Puente, J.-P. Ampuero, G. Brietzke, M. Käser, Three-Dimensional Dynamic Rupture Simulation with a High-order Discontinuous Galerkin Method on Unstructured Tetrahedral Meshes, J. Geophys. Res. - Solid Earth, 2012

[14] *http://www.scalasca.org*

[15] X. Guo, Report on Application Enabling for Capability Science in the MIC Architecture, PRACE Deliverable D7.1.3, *http://www.prace-ri.eu/IMG/pdf/d7.1.3_1ip.pdf*

[16] A. Heinecke et al., Optimised Kernels for Large Scale Earthquake Simulations with SeisSol, an Unstructured ADER-DG Code, poster at SC’13

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-261557. The results were obtained within the PRACE Preparatory Access Type C Project 2010PA1759 “Porting and optimising SeisSol on the Intel MIC architecture”. We thank CINECA for timely and useful assistance.