Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

SHAPE Project Milano Multiphysics:

Evaluation of the Intel Xeon Phi performances for high fidelity nuclear applications

Carlo Fiorina[a*], Giorgio Amati[b], Vittorio Ruggiero[b], Ivan Spisso[c]

[a]*Milano Multiphysics S.R.L.S, Via Giorgio Washington 96, 20146 Milan, Italy*
[b]*SuperCompunting Application and Innovation Department, Cineca, Via dei Tizii, 6 40133, Roma, Italy*
[c]*SuperCompunting Application and Innovation Department, Cineca, Via Magnanelli 6/3, 40133, Casalecchio di Reno, Bologna, Italy*

Abstract

Milano Multiphysics is a start-up active in the field of advanced modelling of complex systems. The goal of this SHAPE project was to evaluate the pros and cons of the Intel's Many Integrated Core technology in the highly demanding field of nuclear engineering. More specifically, the performances of the Intel Xeon Phi processors have been evaluated for two commonly used tools in the nuclear engineering community, namely: the OpenFOAM finite-volume library, and the Serpent Monte Carlo code. Some notable advantages compared to traditional Intel Xeon processors have been observed in terms of both speed-up and energy consumption. However, these advantages are limited to the case of OpenFOAM while Serpent seems to perform worse on Xeon Phi processors, which is partly due to the impossibility to vectorise traditional implementations of Monte Carlo algorithms. Additional margins of improvements for both codes may actually come from a better use of vectorization-friendly algorithms.

1. The company Milano Multiphysics

Milano Multiphysics is a company founded in 2015 by recognized researchers, offering products and services related to the modelling and simulation of complex systems. It builds upon a long experience in the multidisciplinary and highly demanding field of nuclear reactor design and uses it as a leverage to build innovative and reliable solutions for a wide range of engineering applications. Its expertise focuses on highly coupled numerical simulations of complex systems, Monte Carlo analysis, data analysis, uncertainty quantification, method development, numerical implementation, as well as on the related activities of verification and validation, know how development and technology transfer. Within this PRACE SHAPE project, the focus was on the investigation of the use of non-traditional processor architectures. This is a possible mean to ease the computational constraints that affects modern high-fidelity numerical simulations, notably in the field of nuclear engineering.

*Corresponding author. E-mail address: carlo.fiorina@milanomultiphysics.com

2. Introduction

This work originated from the activities of Milano Multiphysics in the field of high-fidelity modelling of nuclear reactors. For this kind of applications, one has to simulate both the thermal-hydraulics of the coolant, and the transport of neutrons in the system.

The OpenFOAM CFD toolbox [1] is an emerging tool in the nuclear engineering community thanks to some attractive features:

- It includes several advanced and verified solvers for single-phase and multi-phase flow;
- Despite its thermal-hydraulics vocation, it has been structured as a general library for the finite-volume discretization and solution of PDEs, thus offering a sound numerical environment for multiphysics applications;
- It supports parallelism, scaling with good performances on modern HPC resources;
- It is based on a cost-free GNU GPL license, which allows for collaborative developments;
- It is based on an effective object-oriented programming style;
- It is based on C++, one of the most widely used programming languages in the scientific community.

These features have triggered many research efforts in the nuclear community, resulting in several tens of publications [2] and leading to the development of highly specialized solvers for reactor thermal-hydraulics [2]; a set of solvers for neutron transport (e.g., [3][4][5][6][7]); and a few full-fledged solvers for the multi-physics analysis of nuclear reactors (e.g., [9][10][11][12]). Following the growing efforts in the field, an international research initiative is being launched for coordinating the various efforts related to the use of OpenFOAM in the nuclear field†. The objective is the creation of an open-source multi-physics platform for reactor analysis.

Despite the clear interest in advanced CFD and multiphysics applications, high-fidelity simulations are extremely demanding and the effective use and development of these tools is closely linked to developments in the HPC domain. Recent developments in this field indicate a growing interest in non-traditional CPU architectures that are developed in the attempt to achieve higher performances while limiting power consumption. In particular, a paradigm shift is proposed: instead of increasing chip performances through a maximization of single thread performances, new chips are designed using a large number of simpler units with poor single thread performances but maximum throughput [13]. Two main approaches are emerging in this sense: the use of NVIDIA/GPU accelerators, and the development by Intel of the Xeon Phi CPU family. The Xeon Phi processors make use of Intel's Many Integrated Core (MIC) technology to use a high number of low-performance cores (60 or plus), each one capable of handling up to 4 threads (with a total of 240 or more threads per processor). In addition, these processors are designed to make an intensive use of vector processing units (two per each core), and of an integrated high-speed memory (MCDRAM). An advantage of Intel Phi over the use of NVIDIA/GPU accelerators is that there is no need to rewrite available solvers in CUDA, as standard programming languages are compatible with the Intel Xeon Phi CPU. This makes the latter suitable for use for available codes.

The main objective of this paper is to compare the performances of OpenFOAM-based solvers on traditional Intel Xeon and on innovative Intel Xeon Phi processors. In addition, preliminary results are provided on the performances of codes based on a Monte Carlo methodology. The latter represents as of today the reference methodology in the nuclear community for the accurate prediction of neutron transport.

3. Computational platform

To achieve its objectives, this work has made use of the MARCONI HPC cluster operated by CINECA [14]. MARCONI is based on the Lenovo NeXtScale platform and builds upon the Intel Xeon Phi product family alongside with Intel Xeon processor product family. The partition A1 (named BDW) is based on "traditional" Intel Xeon processor E5-2600 v4 product family (Broadwell) with a computational power of 2Pflop/s. The partition A2 (named KNL) is instead equipped with the next-generation of the Intel Xeon Phi product family (Knights Landing), with a computational power of ~ 11Pflop/s. In particular, the BDW partition features nodes constituted by 2 x 18-cores Intel Xeon E5-2697 v4 at 2.30 GHz. The KNL partition is instead based on nodes with 1 x 68-cores Intel Xeon Phi 7250 at 1.4 GHz.

---

† *This research project, named foam-for-nuclear, has been launched as a pilot project in the frame of an initiative of the Knowledge Management Section of the International Atomic Energy Agency (https://www.iaea.org/nuclearenergy/nuclearknowledge/strengthening-Nuc-edu/uni-collaboration.html)*

4. Methodology

To investigate the performances of OpenFOAM on the two selected computer architectures, namely Intel Xeon and Intel Xeon Phi, it has first been necessary to single out suitable profiling tools. An attempt has been made to employ the Scalasca profiling tool [15]. However, and thanks to the Score-p [16] development team, a problem was identified in the fact that the PStream library employed by OpenFOAM for parallel communications tends to hide the used MPI infrastructure to Score-p. The result is that while linking the executable, neither the MPI library nor the MPI compiler wrapper appear at the link line. This causes the Score-P MPI library wrapping to fail, even though the analysis of the executable shows that the PStream library contains the Score-P MPI wrapper library as a dependency. Simply adding MPI_Init/MPI_Finalize and the start/end of main, makes Score-P link in MPI mode without errors and allows to run MPI measurement. However, it stills fails to wrap the MPI calls of the PStream OpenFOAM library and, thus, to provide any communication information in the results. Another issue is that OpenFOAM overwrites malloc. This clashes with the Score-P memory measurement system that wraps malloc. Finally, GCC enable Score-P to distinguish between inlined and non-inlined functions, which makes it a preferable compiler when using Score-P for C++ codes. However, the GCC compiler has been proved as unsuited for compiling OpenFOAM on Xeon Phi machines [17].

A first few tests have been run using the Intel MPI Performance Snapshot [18]. However, its use caused a significant overhead in computing time (up to 8 times compared to the computing time without instrumentation), thus limiting the reliability of the measured performances.

Two other tools proved instead to be suited for profiling OpenFOAM, i.e., the HPCToolkit [19] and the Intel toolset: Trace Analyzer, Advisor and VTune Amplifier XE [20]. The Intel tools have been chosen for this work.

5. Performance comparison - OpenFOAM

As a test case, the icoFoam solver available in OpenFOAM has been used to evaluate the flow field in a cubic cavity with a moving wall. Solution time is largely dominated by the solution of the pressure field. This solution will be achieved based on two different algorithms, namely: 1) a preconditioned conjugate gradient algorithm (PCG) with a simplified diagonal-based incomplete Cholesky (DIC) preconditioner; and 2) a Geometric agglomerated algebraic multigrid solver (GAMG) with Gauss-Seidel smoother [21].

5.1 Single-node performances

As a first test, the performances of single nodes of the BDW and KNL partitions of Marconi have been compared. For this case, each side of the simulated cavity is discretized in 200 segments, resulting in a total number of cells equal to 8 millions. Twenty time steps have been simulated. No hyper threading has been used. A number of threads per node equal to 32 has been employed for the BDW partition. Both 32 and 64 threads have instead been used for the KNL partition. Results are summarized in Table 1.

Table 1: Performance comparison between BDW and KNL partitions on a single node

	BDW – Xeon (32 threads)	**KNL - Xeon Phi (32 threads)**	**KNL - Xeon Phi (64 threads)**
PCG	1433	1412	805
GAMG	454	759	335

It is worth noting the significantly better performances of a single KNL node compared to a BDW node. This result is significant if one considers that the cost of a KNL node is essentially half the cost of a BDW node, and that the energy consumption reduces by approximately 25% (evaluated based on the processor Thermal Design Power - TDP).

The performances of the hyper threading technology have also been evaluated. As expected for CFD problems, the use of multiple threads per core provides limited improvements. On KNL, the use of 128 tasks on a node leads to a solution time of 657 seconds, to be compared to the 805 seconds with 64 threads. Use of 512 tasks has instead the effect of reducing the performances, with a solution time of 694 seconds.

5.2 Scaling behaviour

The scaling performances of OpenFOAM on the Xeon and Xeon Phi processors have first been tested based on a traditional strong scaling study, i.e., by increasing the number of cores while maintaining the size of the problem. Also in this case, no hyper threading has been used and a number of threads per node equal to 32 and 64 has been employed for the BDW and the KNL partitions, respectively. Results are shown in Figure 1and Figure 2.

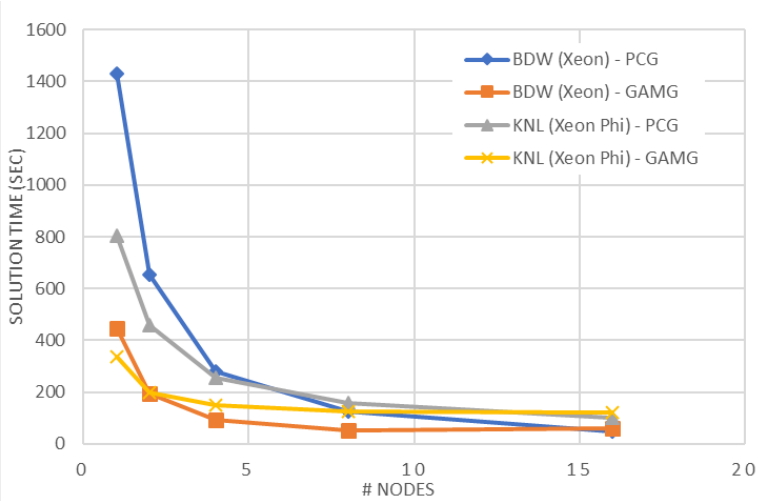

Figure 1: Computing time of the BDW and KNL partitions for a strong scaling study

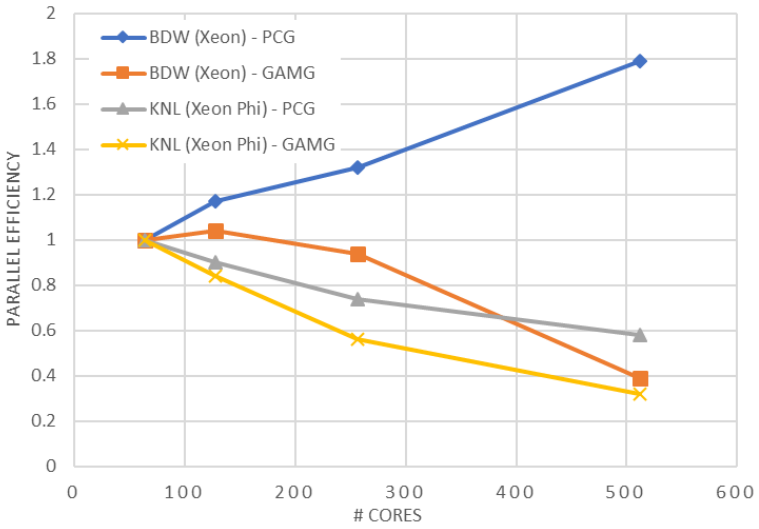

Figure 2: Parallel efficiency of the BDW and KNL partitions for a strong scaling study, using the performances for 64 cores as reference

OpenFOAM shows better scaling performances on BDW vs KNL, suggesting a potentially higher parallel load for the Xeon Phi processors. This has been confirmed via an analysis with the Intel Trace Analyzer tool, which suggests for the 64 cores PCG case a parallel load equal to 7.9% and 9.4% for BDW and KNL, respectively. The worsening of the scaling behaviour when changing from BDW to KNL is more evident for the PCG solver.

To get a better insight into the scaling performances of the BDW and KNL partitions, a weak scaling study has been performed by decomposing the domain into sub-domains of 62500 cells each and by increasing the number

of sub-domains (and cores). The results are reported in Figure 3. The scaling efficiencies are calculated based on the time required for each pressure iteration[‡], assuming the 64 cores case as reference.

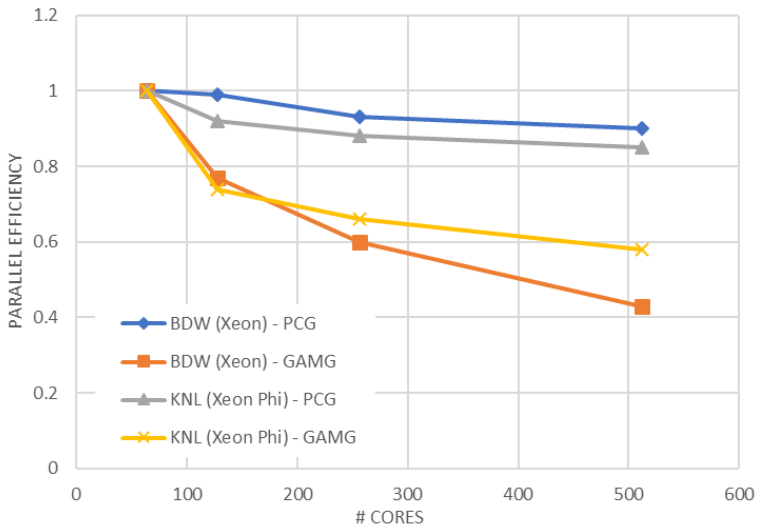

Figure 3: Parallel efficiency of the BDW and KNL partitions for a weak scaling study, using the performances for 64-cores as reference

Weak scaling studies have the advantage of keeping constant the serial load for each thread. In OpenFOAM, the parallel load depends on the ratio between number of processor faces and internal cells, and on a higher probability of load imbalances, which are both expected to grow with increasing number of sub-domains (and cores). Figure 3 shows an interesting trend for the KNL partition, which shows worse efficiencies for the PCG solver, but better efficiencies for the GAMG solver.

To understand the trends observed in Figure 2 and Figure 3, the parallel load of different simulations has been investigated using the Intel Trace Analyzer tool. In particular, the 64 core cases of the weak scaling study have been used. Results are reported in Table 2.

Table 2: Decomposition of the parallel load (%) for the BDW and KNL partitions

	BDW - Xeon		**KNL - Xeon Phi**	
	PCG	**GAMG**	**PCG**	**GAMG**
MPI (allreduce)	66.1	26.8	80.5	52.5
MPI (waitall)	18.7	30.0	7.9	26.4
MPI (send)	8.7	22.7	6.6	11.6
MPI (receive)	6.5	20.5	4.9	9.5

It can be observed that the parallel load in the PCG solver is largely dominated by the “allreduce” parallel operation. Such parallel operation performs worse on KNL, which leads to the worse parallel efficiency observed in Figure 2 and Figure 3. The send and receive operations contributes instead significantly to the parallel load of the GAMG solver. These operations perform significantly better on KNL, thus resulting in the better scaling performances observed in Figure 3. The worse scaling performances observed for KNL and the GAMG solver in Figure 2 are related to the growing weight of the allreduce operation in a strong scaling study. For instance, the fraction of the MPI load associated to the allreduce operation changes from 10.8% to 20% when launching the

[‡] *The necessity to normalize by the number of iterations derive from the notable increase of required iterations with increasing number of sub-domains.*

simulation on 1 and 2 KNL nodes, respectively. In view of the importance of the allreduce operations, different implementations of this algorithms have been tested. The Intel compiler allows in fact for 9 different implementations of this algorithm [22]. However, no improvements have been observed compared to the default option.

5.3 Impact of vectorization

Major advantages in the use of the Xeon Phi is expected for software that can make use of its two 512 bits vector units per core. To preliminary evaluate the effect of vectorization on OpenFOAM solvers, two separate OpenFOAM compilations have been performed, i.e. with and without (flag no-vec) auto-vectorization. The same case as in the previous section has been considered, but with only 100 cells per each side of the cubic cavity. In view of the limited size of the problem, only 8 cores have been employed. The PCG solution algorithm with DIC preconditioning has been used. Results showed that vectorization reduces the computing time from 353.3 to 330.7 seconds, i.e. by less than 10%[§]. Although non-negligible, this improvement can be considered as marginal compared to the potential advantages that arise from the use of multiple large vector processing units. In order to investigate this behaviour, an in-depth analysis has then been carried out using Intel VTune on an instrumented version of OpenFOAM. This allowed to single out the most time-consuming loops and, for each of them, the impact of vectorization. These results are summarized in Table 3.

Table 3: List of the most time-consuming loops and result of vectorization

File location	**Line**	**Code snippet**	**Vectorised**	**Time w/o vectorization (s)**	**Time with vectorization (s)**

src/OpenFOAM/matrices/lduMatrix/preconditioners/DICPreconditioner/DICPreconditioner.C	109	for (label cell=0; cell<nCells; cell++)
		{
		wAPtr[cell] = rDPtr[cell]*rAPtr[cell];
		}
			Yes	9.2 (2.6%)	5.3 (1.6%)

	114	for (label face=0; face<nFaces; face++)
		{
		wAPtr[uPtr[face]] -= rDPtr[uPtr[face]]*upperPtr[face]*wAPtr[lPtr[face]];
		}
			No	59.9 (17.0%)	59.5 (18.0%)

	119	for (label face=nFacesM1; face>=0; face--)
		{
		wAPtr[lPtr[face]] -= rDPtr[lPtr[face]]* upperPtr[face]*wAPtr[uPtr[face]];
		}
			No	66.6 (18.9%)	67.1 (20.3%)

/home/carlo/OpenFOAM/OpenFOAM-v1706/src/OpenFOAM/matrices/lduMatrix/lduMatrix/lduMatrixATmul.C	68	for (label cell=0; cell<nCells; cell++)
		{
		ApsiPtr[cell] = diagPtr[cell]*psiPtr[cell];
		}
			Yes	8.9 (2.5%)	5.7 (1.7%)

	76	for (label face=0; face<nFaces; face++)
		{
		ApsiPtr[uPtr[face]] += lowerPtr[face]*psiPtr[lPtr[face]];
		ApsiPtr[lPtr[face]] += upperPtr[face]*psiPtr[uPtr[face]];
		}
			No	76.5 (21.7%)	77.2 (23.3%)

/home/carlo/OpenFOAM/OpenFOAM-	151	for (label cell=0; cell<nCells; cell++)
		{
		pAPtr[cell] = wAPtr[cell] + beta*pAPtr[cell];
		}
			yes	8.6 (2.4%)	4.5 (1.4%)

[§] *This result has been obtained by comparing two different runs with and without vectorization. The speed-up evaluated by Intel Advisor was estimated to be ~25%, showing a notable overestimation by this tool.*

v1706/src/OpenFOAM/matrices/lduMatrix/solvers/PCG/PCG.C	172	for (label cell=0; cell<nCells; cell++)
		{
		psiPtr[cell] += alpha*pAPtr[cell];
		rAPtr[cell] -= alpha*wAPtr[cell];
		}
			yes	17.2 (4.9%)	8.6 (2.6%)

The most time-consuming loops in the case here analysed cannot vectorise due to the existence of internal dependencies, which explain the generally low impact of vectorization. In addition, the speed-up of the few vectorized loops is limited. This is suspected to be due to the double indexing often used in OpenFOAM (see code snippets in Table 1), which normally leads to significant cache misses and a slower recovery of data for vector operations. Another reason might be the general difficulty of fetching data from the memory that is typical of CFD codes. A test has been run where the #pragma directive has been used to force the vectorization of the loops in the DICPreconditioner.C file. Aside from the obvious consequence of leading to wrong results, it caused the loop at line 114 of DICPreconditioner.C to speed-up by two times, and the one at line 119 of the same file to slow down by two times. This indicates a clear difficulty of the compiler in interpreting loops making use of double indexing.

Some preliminary tests have also been carried out to investigate the impact of vectorization on other OpenFOAM linear system solvers and preconditioners. In all cases advantages of vectorization were found to be limited. In particular, vectorization had no impact on the GAMG solver.

6. Performance comparison for a Monte Carlo code

As mentioned, standard nuclear reactor calculations often involve the use of Monte Carlo codes for neutron transport. To evaluate the potential impact of the Intel Xeon Phi technology in this field, it is then important to quantify its performances for these types of codes.

In this work, the Serpent Monte Carlo code developed at the VTT in Finland [24] has been selected as it represents one of the most widely employed Monte Carlo codes in the nuclear community. To preliminary test its performances on the BDW and KNL partitions of Marconi, a strong scaling study has been performed. As a test case, a simple simulation of a lead-cooled fast reactor has been performed employing 30 million neutrons. A number of cores per node equal to 32 and 128 has been employed for BDW and KNL, respectively. This is equivalent to employing approximately half of the threads that would be potentially available for each node when using the hyper threading technology.

Different from the case of OpenFOAM, Figure 4 shows that the Serpent Monte Carlo code scales linearly for both architectures. On the other hand, the performances of the KNL partition are in this case significantly worse. On a single node, computing time is equal to 21.2 seconds and 51 seconds for BDW and KNL, respectively.

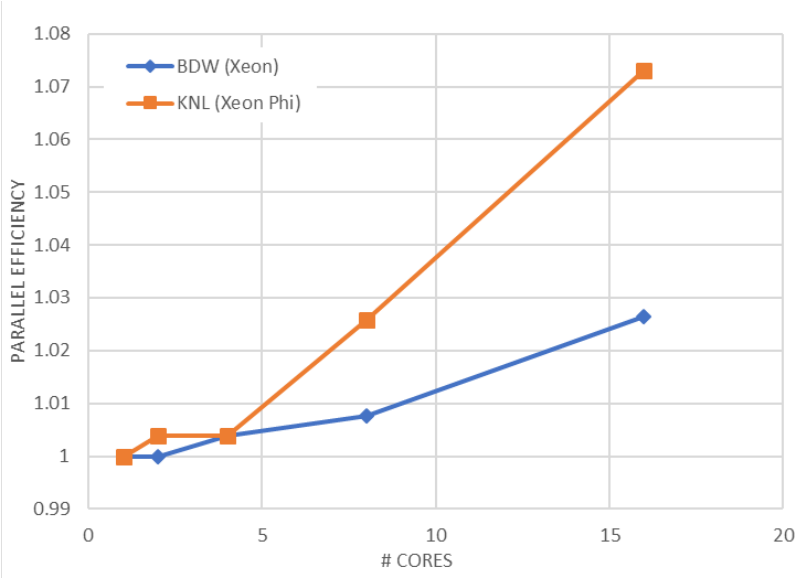

Figure 4: Performance comparison between BDW and KNL for a strong scaling study using Serpent

Monte Carlo codes are characterized by an extremely small exchange of information between threads, which explains the observed (and well known) good scalability. For what concerns the lower performances of KNL vs BDW for this kind of codes, Tramm and Siegel indicate that the single core performances of a Monte Carlo code tends to be communication-bound. In particular, bottlenecks derive from frequent cache misses. The smaller L2 cache of the Intel Xeon Phi 7250 (32 MB) compared to the Intel Xeon E5-2697 v4 at 2.30 GHz (45 MB) can then help explaining the lower performances of the former. Another reason is that traditional implementation of Monte Carlo codes for neutron transport do not allow for vectorization. Specific implementations that could benefit from the presence of vector processing units have been devised [25], but they are not commonly employed as of today.

7. Conclusions

In this paper, the use of the recent Intel Xeon Phi technology has been investigated as a possible mean to tackle the growing computational requirements in the field of nuclear engineering. To this purpose, a comparison has been carried out between the performances of modern tools for reactor analysis when used on an Intel Xeon Phi and on a traditional Intel Xeon architecture. The deterministic finite-volume OpenFOAM code, and the stochastic Serpent code have been tested. As far as the specific node architecture is concerned, the BDW (Intel Xeon) and KNL (Intel Xeon Phi) partitions of the CINECA Marconi cluster have been used. It has been observed that the Xeon Phi technology can provide a notable speed-up for OpenFOAM based solvers up to a few hundreds of MPI threads. This is particularly of interest considering the lower cost (half) and energy consumption (25% less) of a Xeon Phi based node. It was noted that OpenFOAM hardly makes use of the availability in the Intel Phi processor of two 512-bit vector processing units for each core. In fact, the most time-consuming loops in the code cannot be vectorised due to the existence of internal dependencies. In addition, vectorised loops experienced limited acceleration. This has been ascribed to the double indexing of loop variables often employed in OpenFOAM, as well as to the inefficiencies in data retrieval from memory that is typical of CFD codes. In terms of scaling, the Intel Xeon Phi displays an improved behaviour for solvers (like GAMG) heavily relying on simple send and receive operations, and a notably worse behaviour on solvers (like PCG) that make intensive use of the allreduce operation.

As far as Monte Carlo codes are concerned, it has been observed that scaling is essentially linear for both the BDW and the KNL partitions of Marconi, but that the Xeon Phi nodes perform significantly worse compared to the more traditional Xeon nodes. In this sense, it is known and it has been confirmed that current typical implementations of Monte Carlo routines for neutron transport do not make use of vectorization.

To conclude, notable potential advantages have been observed in the use of Intel Xeon Phi processors, including a lower cost, lower energy consumption per node, and notably improved performances for OpenFOAM-based solvers. However, significantly worse performances have been observed for Monte Carlo codes. Both OpenFOAM and Monte Carlo codes make little or no use of the vector processing units of the Intel Xeon Phi technology, which indicates that “vector-friendly” implementations of these codes may result in dramatically improved performances of the Xeon Phi processor family compared to the standard Xeon Phi.

Based on these results Milano Multiphysics is planning to pursue its investigation and use of the Intel’s Many Integrated Core technology with the objective of pushing forward the current limit of feasibility for industrial calculations in the field of nuclear engineering. The possibility will also be considered to initiate some R&D activities dedicated to the development of vector-friendly algorithms for CFD and Monte Carlo methodologies.

References

[1] OpenFOAM, 2018. http://www.openfoam.com/ , http://www.openfoam.org/.

[2] SIG Nuclear, 2017. OpenFOAM nuclear special interest group. https://openfoamwiki.net/index.php/SIG_Nuclear_Simulations.

[3] Aufiero, M., 2014. Development of advanced simulation tools for circulating fuel nuclear reactors. PhD Thesis. Politecnico di Milano, Italy.

[4] Jareteg, K., Vinai, P., Demazière, C., 2014. Fine-mesh deterministic modeling of PWR fuel assemblies: Proof-of-principle of coupled neutronic/thermal–hydraulic calculation. Annals of Nuclear Energy 68, 247-256.

[5] Jareteg, K., Vinai, P., Sasic, S., Demazière, C., 2015. Coupled fine-mesh neutronics and thermal-hydraulics – Modeling and implementation for PWR fuel assemblies. Annals of Nuclear Energy 84, 244-257.

[6] Fiorina, C., Kerkar, N., Mikityuk, K., Pautz, A., 2016. Development and verification of the neutron diffusion solver for the GeN-Foam multi-physics platform. Annals of Nuclear Energy 96, 212-222.

[7] Fiorina, C., Hursin, M., Pautz, A., 2017. Extension of the GeN-Foam neutronic solver to SP3 analysis and application to the CROCUS experimental reactor. Annals of Nuclear Energy 101, 419-428.
[8] Aufiero, M., Cammi, A., Geoffroy, O., Losa, M., Luzzi, L., Ricotti, M.E., Rouch, H., 2014. Development of an OpenFOAM model for the Molten Salt Fast Reactor transient analysis. Chemical Engineering Science 111, 390–401.
[9] Clifford, I., 2013. A hybrid coarse and fine mesh solution method for prismatic high temperature gas-cooled reactor thermal-fluid analysis. PhD Thesis. PennState University, US.
[10] Clifford, I., Ivanov, K.N., Avramova, M.N., 2013. A multi-scale homogenization and reconstruction approach for solid material temperature calculations in prismatic high temperature reactor cores. Nuclear Engineering and Design 256, 1-13.
[11] Fiorina, C., Mikityuk, K., 2015. Application of the new GeN-Foam multi-physics solver to the European Sodium Fast Reactor and verification against available codes. ICAPP 2015 Conference, May 03-06, Nice, France.
[12] Fiorina, C., Clifford, I.D., Aufiero, M., Mikityuk, K., 2015. GeN-Foam: a novel OpenFOAM® based multi-physics solver for 2D/3D transient analysis of nuclear reactors. Nuclear Engineering and Design 294, 24-37.
[13] Emerson, A., Presentation at Cineca Winter School 2017.
[14] CINECA, 2017. http://www.hpc.cineca.it/hardware/marconi .
[15] Scalasca, 2017, http://www.scalasca.org/ .Gaston, D., Newman, C., Hansen, G., Lebrun-Grandié, D., 2009.
[16] Score-p, 2017a, http://www.vi-hps.org/projects/score-p/ .
[17] Score-p, 2017b. https://trac.version.fz-juelich.de/vis/wiki/Software/OpenFOAM/config_ScoreP
[18] Intel, 2017a, https://software.intel.com/sites/products/snapshots/application-snapshot/ .
[19] HPC Toolkit, 2017, hpctoolkit.org .
[20] Intel, 2017b, https://software.intel.com/en-us/tools-by-segment/technical-enterprise .
[21] Quarteroni, A., Sacco, R., Saleri, F., 2007. Numerical Mathematics. Second Edition. Springer Berlin Heidelberg New York.
[22] Wesseling, P., Oosterlee, C.W., 2001. Geometric multigrid with applications to computational fluid dynamics. Journal of Computational and Applied Mathematics 128, 311-334.
[23] Intel, 2017c. https://software.intel.com/en-us/articles/intel-mpi-library-collective-optimization-on-intel-xeon-phi.
[24] Leppänen, J., 2007. Development of a New Monte Carlo Reactor Physics Code. D.Sc. Thesis, Helsinki University of Technology, Finland.
[25] Tramm, J.R., Siegel, A.R., 2013. Memory Bottlenecks and Memory Contention in Multi-Core Monte Carlo Transport Codes. Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013), La Cité des Sciences et de l'Industrie, Paris, France, October 27–31, 2013.
[26] Romano and Siegel, 2018. Limits on the efficiency of event-based algorithms for Monte Carlo neutron transport. Nuclear Engineering and Technology 49, 1165-1171.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's Horizon 2020 Research and Innovation programme (2014-2020) under grant agreement 730913.