Journal of Physics: Conference Series **513** (2014) 052024 doi:10.1088/1742-6596/513/5/052024

Does the Intel Xeon Phi processor fit HEP workloads?

A Nowak, G Bitzes, A Dotti, A Lazzaro, S Jarp, P Szostek, L Valsan, M Botezatu, J Leduc

Andrzej.Nowak@cern.ch, Georgios.Bitzes@cern.ch, Sverre.Jarp@cern.ch, Pawel.Szostek@cern.ch (CERN openlab, Geneva, Switzerland)

Abstract. This paper summarizes the five years of CERN openlab's efforts focused on the Intel Xeon Phi co-processor, from the time of its inception to public release. We consider the architecture of the device vis a vis the characteristics of HEP software and identify key opportunities for HEP processing, as well as scaling limitations. We report on improvements and speedups linked to parallelization and vectorization on benchmarks involving software frameworks such as Geant4 and ROOT. Finally, we extrapolate current software and hardware trends and project them onto accelerators of the future, with the specifics of offline and online HEP processing in mind.

1. Introduction

The High Energy Physics (HEP) community has always wanted "unlimited" amounts of CPU power for its computing, in particular for data mining and simulation, which are two of the main computing tasks linked to an experiment. Online triggering and offline reconstruction, the two other main tasks, also require a lot of processing but are more tied directly to the number of events produced by the online trigger function.

The FORTRAN programs used to be relatively limited in size – 10'000 lines was the standard at the time of Rubbia's Nobel-Prize-winning experiment in the early 80s. Today, the programs have developed into complex C++ software frameworks representing millions of lines. The most compute-intensive parts depend on efficient execution on control-flow-based sections. Take the example of simulation. The control-flow varies constantly based on random numbers. Is the current particle an electron? Is it inside liquid argon (as material) and a polycone (as geometric shape)? Is the particle about to annihilate and produce other particles? A simulation program moves "forward" randomly based on the values of random numbers used to decide what kind of physics processes are allowed to happen.

In the search for the most cost-effective solution, HEP moved from supercomputer and mainframes (in the 80s) to powerful RISC workstations and servers in the 90s. In the mid-90s PC technology had matured considerably. This style of computing has lasted until today and the Worldwide LHC Computing Grid (WLCG) is almost exclusively based on two-socket commodity PC servers interconnected via high-speed (1Gb and 10Gb Ethernet).

In spite of this total reliance on commodity technology (even for world-class supercomputers) some high performance users started to move to accelerators, in particular to GPGPUs. Initially the emphasis was made on single-precision (SP) floating point, which is more than good enough for most graphics applications and some scientific ones, such as those from the financial world, but not for the common HEP applications which overwhelmingly depend on DP. The reasons are partly historical, but partly related to the complexity of existing algorithm implementations, which cannot be

sufficiently verified in single precision – even today some are advising a move to quad precision. Many prototypes of simpler algorithms exist in single precision, and are slowly spreading through the community. Things gradually changed when more and more graphics cards started appearing with acceptable DP performance, which could run existing software without major modifications. This fact led the CERN openlab team collaborating with Intel to prick up their ears when it was known (in 2008) that Intel was working on an accelerator card, named "Larrabee". The obvious question was whether a new move was in preparation.

2. The history of Intel MIC at CERN openlab

"Larrabee" was initially discussed with Intel Labs, who expressed the hope that several sciences would find this architecture promising, given its flexibility. The many-core architecture came with a promise of simpler, individual cores with higher throughput per mm^2 (or watt). Although the CPU itself was rather ancient (based on a Pentium design, the P54C), a brand-new vector design had been introduced. In spite of the fact that HEP had not had much success with vectors in the past, the flexibility of this new design was intriguing: a large number of vector registers as well as separate mask registers for optimizing different control-flow for each vector element.

Simulator runs as well as a comprehensive review of the architecture helped steer towards the needs of HEP programs. Our review put the finger on HEP's need for good DP floating-point in general and for good hardware support for several important mathematical functions, such as square root, but also exponentials, logarithms, and several trigonometric functions.

The first card appeared in openlab in 2009 and represented one of the first cards to be used outside of Intel and the world of graphics. Shortly later, the name was changed to "Knights Ferry" (KNF) and the decision had been made to put the emphasis on high-performance (and high-throughput) computing. The project was still very secret at this point.

Ever since its inception the operating system on the card was FreeBSD, but it was immediately clear to the openlab team that a Linux version would be much more practical. The request was quickly made to Intel and a year later they started shipping the cards with an adapted version of Linux. Since KNF was still heavily biased towards SP floating-point the decision was made to port a Trackfitting code that was developed by the CBM community [2][3] for the CBM experiment, and also usable by other heavy-ion experiments, such as ALICE in LHC. This port was very successful (see Table 2) and led to a customer testimonial in 2010 at the ISC conference where Intel made the world-wide announcement of the commercial availability of these (now called) co-processors in the form of a "Knights" family, also called the "Many Integrated Cores" (MIC) architecture.

2nd generation "Knights Corner" (KNC) cards started appearing in openlab about a year ago and form the hardware basis for this paper. What has to be kept in mind is that the results reflect a certain number of limitations still inherent in the initial design, as seen from the viewpoint of HEP. The micro-architecture is still based on the Pentium P54C (in-order) design and there is little main memory for multiple processes. The next version of the family, "Knights Landing" or KNL, will lift all these limitations and we are therefore optimistic that the work performed in CERN openlab, jointly with the LHC experiments and the EU Marie Curie Actions FP7 "ICE-DIP" project, will pay off past 2015.

3. The architecture

Like its predecessor, the KNC carries a number of changes with respect to mainstream architectures. These are briefly discussed in the sub-sections that follow.

3.1. Core architecture and count

A single KNC core is based on an in-order, 2-way superscalar x86 architecture codenamed P54C, derived from the Intel Pentium series of processors. It features a range of improvements discussed below. Clock frequencies of production models vary between 1.05-1.25 GHz. Each core features 32kB data and instruction L1 caches, and access to a large, coherent, shared L2, in which each core has 512kB assigned. Loads to L2 are hardware prefetched, but loads to L1 are not and may have to be

handled by software prefetch instructions on some workloads to avoid frequent L2 accesses. A bi-directional 512-bit ring bus interconnects the cores. A similar design also interconnects cores and the GPU unit on newer mainstream Xeon models.

A further modification to take note of is the issue unit – a single hardware thread can issue instructions only every other cycle, but a single core can issue an instruction every cycle. Therefore at least two hardware threads are needed to keep a core busy every cycle.

An interesting development is the core count, which reaches 61 in top-bin KNC models. This architectural feature alone takes the device straight into the many-core domain and imposes strict parallelism requirements on prospective workloads.

3.2. Hardware multi-threading

Another major improvement over the classic P54C core is the addition of hardware threading. The KNC core is the first x86 design to be 4-way multi-threaded – a decision that is a consequence of a design that requires multiple simultaneously running threads to be throughput efficient. Unlike GPUs, however, these hardware threads can carry entirely separate streams of execution without performance penalties. The implementation is similar to mainstream x86 Xeon processors, where some on-chip resources are statically partitioned, some are dynamically shared and some are replicated between threads. Overall, top-bin models can run a total of 244 hardware threads.

3.3. Vector units

In parallel to scalar units and registers, new vector units and registers were added, capable of handling 512-bit vectors. Standard data types include 64-bit doubles (8 per vector), 32-bit floats (16) and 32-bit integers (16). Overall, the vector architecture is augmented with respect to mainstream Intel processors, with support for masking/predication (8 registers), 3-way operands, gathers, scatters, permutes and fused multiply-add. The main advantage of masking can be experienced in control flow statements, since upon encountering a masked element the processing units will not read the data source, perform any operation or write to the destination.

As in other architectures, unaligned vector loads will require two instructions and should be avoided. In addition, not all mathematical functions are fully implemented in hardware, and not all are equally supported in single and double precision variants. Table 1, shown by Victor Lee from Intel at IDF2013, shows the throughput of several math functions. The lack of fast support for functions like log, exp or atan2 has implications for HEP code, as measured on an early multi-threaded Geant4 prototype (Figure 1), where they consumed 80% of math time – much more than on a Xeon processor.

Elementary Functions	Tput # of cycles per set of output	Derived Functions	Tput # of cycles per set ofoutput
RECIP	1	DIV	2
RSQRT	1	SQRT	2
EXP2	2	POW	4
LOG2	1	EXP	2
		LOG	2

Table 1. KNC hardware support for math functions.

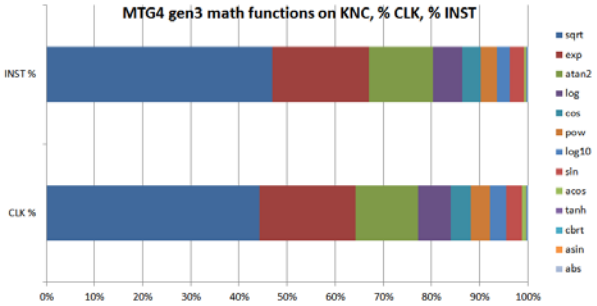

Figure 1. KNC math function profile in an early multi-threaded Geant4 prototype.

3.4. Memory and platform interface

"Knights Corner" exists only in a PCI form factor, which imposes known PCI limitations encountered with other accelerators, such as GPUs and FPGA cards. Memory is implemented through on-board GDDR5, limited to 16GB in the most advanced models. Host memory can be mapped and accessed as well, although with the usual PCI penalties.

4. Programmability and software

4.1. System environment, porting, compilers and software

Internally, the KNC runs a simplified on-board Linux distribution, whose system files are accessible to the user only in a limited manner. KNC devices can run binaries in two modes: either native, where the binary runs directly on the card, or offload, where the binary runs on the host and parts of the computation are offloaded to the co-processor. Binaries compiled for standard x86 systems will not run directly. Recent compilers from Intel (13.1 and later) feature the KNC as a target for cross-compilation. The GNU Compiler Collection is not available by default or as a package, but there are efforts to port it to the co-processor.

It should be noted that many open source packages are not ready for cross-compilation, which sometimes makes porting a complicated business. Such is for example the case of the SPEC benchmark suite, which relies heavily on perl and a range of other dependencies. That was also the case with early multi-threaded Geant4 prototypes, which depended on several packages that were not ready for porting. A later port using newer software packages went much more smoothly, albeit with the usual caveats of cross-compilation still present. Table 2 shows the approximate time related to the porting of several workloads, the evolution of which is discussed in this paper.

Table 2. Approximate porting and tuning times by benchmark.

	LOC	**1st port time**	**New ports**	**Tuning**
Track Fitter (TF)	< 1’000	days	N/A	2 weeks
Maximum Likelihood Fit (MLFit)	3’000	< 1 day	< 1 day	weeks
Multi-threaded Geant4 prototype (MTG4)	2’000’000	1 month	< 1 day	< 1 week

In ideal scenarios, it was sufficient to adapt the workload to the peculiarities of the Intel compiler. In less fortunate cases, it was necessary to either re-implement architecture-specific elements of the workload – such as pinning, vectorization, cache management – or to revise the build system.

4.2. A note on programming languages

On the one hand, the KNC, being an x86-based system, supports a wide range of commonly used programming languages, in particular C, C++ and Fortran. Mainstream compiler support also enables many commonly used technologies and runtimes such as OpenMP, MPI, TBB and Cilk+. On the other hand, commonly used idioms or language features might not provide optimal performance because of architectural differences. The programmer must therefore worry about adapting software to the underlying architecture, in particular in cases where language abstractions obscure architectural peculiarities of the platform.

Subjectively, the Intel MIC and its software ecosystem seem to provide greater accessibility to performance than other accelerators and co-processors, but much work still remains to be done on this front so that the average programmer can have all on-chip transistors within reach.

4.3. Datacenter integration

Current KNC devices feature an Ethernet over PCI interface, as well as libraries for direct messaging between KNC cards and the host, also implemented over PCI. By default, the device appears as a private IP interface visible from the host. The Ethernet bridge eases integration into the datacenter, allowing standard IP-based services to work. As a result, the KNC can access NFS-based filesystems, where workloads can be stored without consuming on-board GDDR5 memory through a virtual filesystem (as was the case in the “Knights Ferry”). Finally, there are no openly available platform management features other than power consumption monitoring.

5. Benchmarks and performance studies

Since 2008, openlab co-developed and ported a range of standard physics benchmarks to the "Knights" platforms. Originally, these were three representatives of HEP programs, covering Simulation (the Multi-Threaded Geant4 prototype), Analysis (MLFit/RooFit), and Online Processing (ALICE/CBM trackfitter). In recent years, as the platform became available to more and more users, openlab collaborators had the opportunity to develop new workloads or updated versions of previously used benchmarks. In the sub-sections below we report on the results of performance studies with such workloads.

The KNC card used was a production unit equipped with 61 cores running at 1.24GHz, with 16 GB GDDR5 memory. The compiler used was ICC 14.0.0, except where specified otherwise.

5.1. HEPSPEC06

Derived from the industry standard SPEC CPU2006 benchmark, the well-known HEPSPEC06 is a HEP-specific benchmark restricted to a single-threaded, non-vectorized C++ subset of SPECINT and SPECFP. Overall, clearly a bad fit for the MIC architecture (and hopeless for GPUs), but representative of many complex HEP workloads. Since the benchmark is single threaded, it must be run in one instance per core and consumes considerable amounts of memory. On the KNC system, it was not possible to exceed 32 simultaneous instances. In addition, we found irregularities in the soplex program, and therefore did not include its results in the geometric mean reported. The compiler used was a MIC-enabled branch of GCC 4.7.

Compiling HEPSPEC06 for MIC required a particular effort. The benchmark is delivered with a set of tools that have to be built before the actual compilation of workloads begins. Our first approach was to cross-compile the whole software package and then perform execution on the card. Since the build script is carrying out validation tests on the resulting binaries, it was not possible to finish this process without introducing some changes. The second approach, more demanding at first sight, was to move the whole build process onto the card. This in turn forced the usage of a MIC-enabled branch of GCC, as well as the compilation of the whole GNU toolchain for the Intel Xeon Phi architecture. A few adjustments in the build scripts and environmental variables were still needed.

The HEPSPEC06 score per KNC core (excluding soplex) was measured to be between 1.75 and 1.80, with linear scaling throughout. A core fully loaded with four hardware threads yields a score of 3.48. The total score for 32 cores was 57.7, which would extrapolate to approximately 110 at a full load of 61 cores. Continuing this extrapolation, one could expect 244 threads to achieve a score of approximately 140-210. This result can be roughly compared to a score of 119 on a 3.1 GHz quad-core Haswell workstation, obtained using GCC 4.8.1. In conclusion, it is a good result, especially considering that SPEC workloads (modeling production HEP software [4]) are not particularly fit for vectorization.

5.2. Next-generation multi-threaded Geant4 prototype (Simulation)

In this test we ran Geant4 [5][6], version 9.6-ref09a (released end of September 2013), which is an internal development version of Geant4 in preparation for version 10.0 (to be released in December 2013). The next major release will include event-level parallelism via multi-threading [7]: after the geometry and physics processes have been initialized, threads are spawned and events are simulated in parallel. To reduce memory footprint the read-only memory parts of the simulation are shared among threads. Memory increase for each additional thread has been measured to be in the order of 40-70 MB, depending on the application – as reported elsewhere at CHEP 2013.

We used the "ParFullCMS" application as a benchmark, which has been developed by the Geant4 collaboration and earlier optimized by openlab. The application uses realistic CMS experiment geometry expressed in GDML format [8], where highly energetic (50 GeV) negatively charged pions are shot in random directions inside the detector. The recommended HEP physics list (FTFP_BERT) is used for simulating the interaction of particles with matter. Tracking in a solenoidal magnetic field

is also included. No digitization of energy deposits is performed since this strongly depends on the experimental framework software and we are interested in evaluating the performance of Geant4 itself.

Performance is measured by keeping the workload of each thread constant (weak scaling) and measuring the time they spend in the event loop. Results are preliminary.

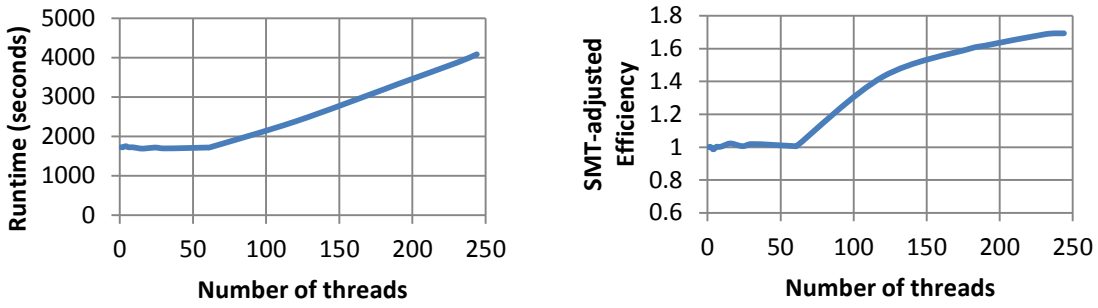

Figure 2: Parallel Geant4 prototype scalability on KNC (preliminary results)

Number of threads	**1**	**2**	**4**	**8**	**10**	**12**	**16**	**24**	**30**	**60**	**120**	**180**	**240**
Events/second	0.06	0.12	0.23	0.46	0.58	0.70	0.95	1.40	1.77	3.49	5.04	5.65	5.88
Scaling	1.0	2.0	3.9	8.0	10.1	12.2	16.4	24.1	30.6	60.4	87.1	97.6	101.6

Table 3: Parallel Geant4 prototype scalability on KNC (preliminary results)

The runtime remains constant as we increase the number of threads (and thus the workload) until we reach the number of physical cores (61), which demonstrates 100% efficiency as the workload increases with the number of threads (Figure 2 and Table 3). With all 244 logical processors being active, the total throughput reached 5.9 events per second. This results is a little bit better than a single-socket, 8-thread "Haswell" workstation at 3.1 GHz, nearly equivalent to a dual-socket 16-thread "Nehalem" platform at 2.26GHz, but less than half of the performance of a dual-socket "Ivy Bridge" system at 2.4 GHz with 48 threads. SMT provided a 69% improvement in throughput.

5.3. Parallel MLFit and the modernized MLFit kernel (Analysis)

The fundamental benchmark used in this test is a vectorized and threaded prototype of the RooFit package from ROOT [9], commonly used in HEP for maximum likelihood fits. It has been described on several occasions by Alfio Lazzaro, in particular in [10] and [11]. In its calculations, the workload makes heavy use of the exp() function and is memory-bandwidth intensive. Speedup measured on the KNC is presented in Table 4.

Table 4: MLFit speedup on KNC

Number of threads	**1**	**2**	**4**	**8**	**16**	**30**	**60**	**119**	**180**	**240**
Speedup	1.00	1.99	3.80	7.41	14.39	23.88	44.06	55.18	55.48	55.72
Runtime	566.6	284.7	149.1	76.5	39.4	23.7	12.9	10.3	10.2	10.2

Internal data block size (with a standard blocking approach) has a considerable impact on performance, with large blocks (10'000) performing 15% better than smaller ones (1'000) – the results were gathered using the better performing value. Vectorization (vs. no-vec) provides a benefit of 4.8x on a single core, which diminishes to 2.7x under a full 240 thread load. The results for SMT scaling and an absolute architecture comparison (with vectorization) are shown in Figure 3 and Figure 4, respectively. In terms of absolute performance on this workload, the KNC was roughly comparable to a dual socket "Ivy Bridge" server with 2x12 physical cores running at 2.4 GHz. Given the similarity in efficiency, a symmetric processing model would likely be more appropriate for such workloads.

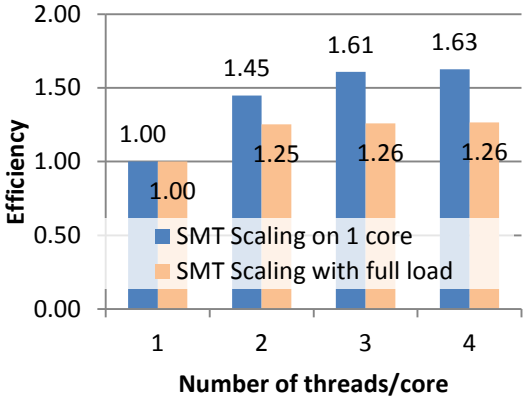

Figure 3: MLFit SMT scaling on KNC (higher is better)

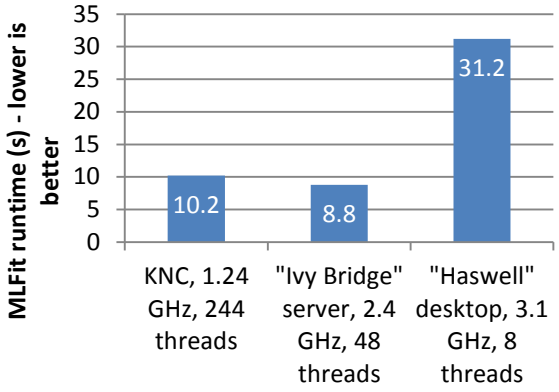

Figure 4: MLFit runtime on three x86 systems in seconds; SMT on; lower is better

A kernel of the MLFit application has been extracted by Vincenzo Innocente from CERN, who applied a series of advanced optimizations, successfully challenging the original tradeoffs and assumptions. While more detail of this work is described in [12], the benchmark exhibits improved scalability, reaching over 90x in a "sweet spot" around 180 threads, as shown in Figure 5.

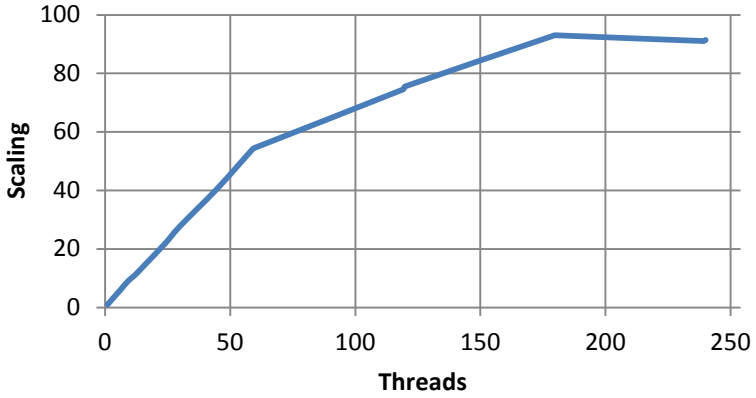

Figure 5: MLFit kernel scaling on KNC

5.4. The evolved Trackfitter (online workload)

Since 2008, openlab has been collaborating with the GSI institute on a novel track fitting benchmark for the ALICE experiment. The workload has been since modernized by Ivan Kisel, Igor Kulakov and Maksym Zyzak of GSI [2][3] and ported to the Vc library developed by Matthias Kretz [13]. The workload is a vectorized Kalman filter, used in a track fitting algorithm. The results for a modernized version of the benchmark using OpenMP are reported in Table 6. The speedup between a double precision x87 variant and a single precision packed variant reached a staggering 14x, close to the ideal 16x. In a direct non frequency-scaled comparison, the KNC card tested performed 11% better than a standard dual socket "Ivy Bridge" server running at 2.4 GHz and 3.5x better than a "Haswell" based desktop running at 3.1 GHz.

5.5. Performance – observations and summary

Performance numbers obtained demonstrate that in order to reach competitive throughput, two conditions must be met: the workload must be threaded and it must be vectorized. Without vectorization and a move from double to single precision, a factor of 14x in performance can be left on the table (see Table 5).

Table 5: Overview of results: [us] or speedup (KNC measurements)

x87	singleVc	OpenMP 213 threads (max)	SP vector to DP scalar speedup	OpenMP to SP vector speedup	OpenMP to DP scalar speedup
11.587	0.811	0.0098	14.2x	82.7x	1182x

Table 6: Real fit time per track [us] – scalability by thread count (rounded to the last presented digit)

1	2	4	8	16	32	48	61	91	122	152	183	213	244
0.812	0.413	0.207	0.104	0.052	0.026	0.018	0.014	0.012	0.010	0.010	0.010	0.010	0.010

At the same time, for vectorization to work well, the issue units should be kept busy every cycle, which implies running at least two threads per core. Benefits of the 3rd and 4th hardware thread were minor in most cases, and more pronounced in the case of workloads bound by instruction flow and branches as opposed to data flow.

6. Conclusions

Several of openlab's suggestions have made it into the final KNC product: amongst others, native mode is a key feature, compilers and important math functions have noticeably improved, system connectivity is now very good, the OS changed to Linux and the OSS ecosystem is growing. All of the workloads initially co-developed have been contributed to the community and, as a result, have seen further evolution steps. The story does not end here, however. Various groups, such as the HEP Concurrency Forum, have taken vectorization in their sights and are actively progressing to build next-generation prototypes of HEP software, such as Geant-V.

It will be very interesting to follow the evolution of the "Knights" family devices. It is already apparent that next-generation units will feature a new core and separate socket options (which was another strong openlab suggestion), which will improve on one of the greatest weaknesses of the Intel MIC: accessibility. CERN openlab will continue to actively collaborate with the physics community on workload optimization and development, and will keep actively monitoring developments in the hardware domain.

7. Acknowledgements

We thank our colleagues for the extensive help they provided us with during our work with the prototypes and production units of the MIC co-processors. We thank Xin Dong, John Apostolakis and Gene Cooperman for their support of the multi-threaded Geant4 prototype. We thank Vincenzo Innocente for a kernelized variant of the MLFit workload. We thank Ivan Kisel, Maksim Zyzak, Igor Kulakov and Volker Lindenstruth from GSI for their work on the ALICE/CBM trackfitter.

Last but not least, we are indebted to our Intel colleagues for their continued, long-term support: Klaus-Dieter Oertel, Jeff Arnold, Hans Pabst, Georg Zitzlsberger, Hans-Joachim Plum and Ralf Ratering from the engineering side, Joe Curley, Herbert Cornelius and Claudio Bellini from the business side, Pradeep Dubey and Victor Lee from the Parallel Computing Lab, as well as many others that we had the privilege to work with over the past 5 years.

Finally, we thank the EU FP7 "ICE-DIP" project, #316596, for support in disseminating this work.

References

[1] Jarp S, Simmins A, Yaari R and Tang H 1995 *PC as physics computer for LHC?*

[2] Kisel I, Kulakov I and Zyzak M Parallel Implementation of the KFParticle Vertexing Package for the CBM and ALICE Experiments *Computing in High Energy and Nuclear Physics 2012*

[3] Kisel I, Kulakov I and Zyzak M Parallel Algorithms for Track Reconstruction in the CBM Experiment Computing in High Energy and Nuclear Physics 2012

[4] Nowak A, Jarp S and Lazzaro A 2012 The future of commodity computing and many-core versus the interests of HEP software *J. Phys. Conf. Ser.* **396** 052058
[5] Agostinelli S, Allison J, Amako K, Apostolakis J, Araujo H, et al 2003 Geant4—a simulation toolkit *Nucl. Instruments Methods Phys. Res. Sect. Accel. Spectrometers Detect. Assoc. Equip.* **506** 250–303
[6] Allison J, Amako K, Apostolakis J, Araujo H, Dubois P A, et al 2006 Geant4 developments and applications *Nucl. Sci. IEEE Trans.* **53** 270–8
[7] Dong X, Cooperman G and Apostolakis J 2010 Multithreaded Geant4: Semi-automatic Transformation into Scalable Thread-Parallel Software *Euro-Par 2010 - Parallel Processing* Lecture Notes in Computer Science ed P D'Ambra, M Guarracino and D Talia (Springer Berlin Heidelberg) pp 287–303
[8] Chytracek R, Mccormick J, Pokorski W and Santin G 2006 Geometry Description Markup Language for Physics Simulation and Analysis Applications *IEEE Trans. Nucl. Sci.* **53** 2892–6
[9] Brun R and Rademakers F 1997 ROOT — An object oriented data analysis framework *Nucl. Instruments Methods Phys. Res. Sect. Accel. Spectrometers Detect. Assoc. Equip.* **389** 81–6
[10] Lazzaro A 2011 Implementing parallel algorithms for data analysis in ROOT/RooFit (Workshop on Future Computing in Particle Physics 2011)
[11] Jarp S, Lazzaro A, Nowak A and Leduc J 2012 *Evaluation of the Intel Sandy Bridge-EP server processor* (http://cds.cern.ch/record/1434748/files/CERN_openlab-Evaluation_of_the_Intel_Sandy_Bridge-EP_server%20processor.pdf)
[12] Innocente V 2013 Concurrency in the minimization of unbinned Log-Likehood (Forum on Concurrent Programming Models and Frameworks)
[13] Kretz M and Lindenstruth V 2012 Vc: A C++ library for explicit vectorization *Softw. Pr. Exp.* **42** 1409–30