




Comparison of HPC Architectures for Computing All-Pairs Shortest Paths. Intel Xeon Phi KNL vs NVIDIA Pascal

Manuel Costanzo¹, Enzo Rucci¹*, Ulises Costi², Franco Chichizola¹, and
Marcelo Naiouf¹

¹ III-LIDI, Facultad de Informática, UNLP – CIC.

La Plata (1900), Bs As, Argentina

{mcostanzo,erucci,francoch,mnaiouf}@lidi.info.unlp.edu.ar

² Facultad de Informática, UNLP.

La Plata (1900), Bs As, Argentina

ulises_92_lp@hotmail.com

Abstract. Today, one of the main challenges for high-performance computing systems is to improve their performance by keeping energy consumption at acceptable levels. In this context, a consolidated strategy consists of using accelerators such as GPUs or many-core Intel Xeon Phi processors. In this work, devices of the NVIDIA Pascal and Intel Xeon Phi Knights Landing architectures are described and compared. Selecting the Floyd-Warshall algorithm as a representative case of graph and memory-bound applications, optimized implementations were developed to analyze and compare performance and energy efficiency on both devices. As it was expected, Xeon Phi showed superior when considering double-precision data. However, contrary to what was considered in our preliminary analysis, it was found that the performance and energy efficiency of both devices were comparable using single-precision datatype.

Keywords: Shortest paths · Floyd-Warshall · Xeon Phi · Knights Landing · NVIDIA Pascal · Titan X

The final authenticated version is available online at https://doi.org/10.1007/978-3-030-75836-3_3

* Corresponding author.

1 Introduction

In the last decade, the quest to improve the energy efficiency of high-performance computing (HPC) systems has fueled the trend toward heterogeneous computing and massively parallel architectures [7]. Heterogeneous systems combine CPUs with accelerators (such as NVIDIA and AMD GPUs or Intel’s Xeon Phi many-core co-processors), delegating code sections with high computational demand to them. According to the Green500 ³ ranking, the November 2010 edition featured 17 systems that integrated accelerators. However, 10 years later, this number has increased to 145, evidencing the great popularity of this strategy.

Today, GPUs can be considered the dominant accelerator class due to their high computing power and energy efficiency, in addition to their low cost. In the opposite sense, one of their weaknesses is the need to learn a specific language in order to make the most of them, such as CUDA and OpenCL. Among the manufacturing companies, NVIDIA stands out as the largest provider for the high-performance segment.

On the other hand, Intel introduced the second generation of its Xeon Phi processors in 2016, codenamed Knights Landing (KNL). Unlike its predecessor, Knights Corner (KNC), KNL can operate as a standalone processor. Among its main features, the large number of cores with hyper-threading support, the incorporation of AVX-512’s 512-bit vector instructions, and the integration of a high-bandwidth memory (HBM), among others [14], can be mentioned. Generations aside, the outstanding feature in this family is that it offers support for x86 architectures, which allows programmers to use traditional models in HPC, such as OpenMP and MPI.

Because each accelerator has its advantages and disadvantages for certain classes of problems [22,3,17], selecting the best option for a given application is key when searching for maximum performance. To provide some guidelines for such selection, this article presents a comparative analysis between two different HPC architectures (Intel Xeon Phi KNL vs. NVIDIA Pascal). As a case study, the Floyd-Warshall (FW) algorithm was selected for computing the all-pairs shortest paths in a graph, as a representative case of graph applications that are memory-bound [16]. We hope that development teams will find this analysis useful when choosing the most suitable architecture for their applications.

The remaining sections of this article are organized as follows: in Section 2, the general background and other works that are related to this research are presented. Next, in Section 3, the implementations used are described, and in Section 4, the experimental work carried out is detailed and the results obtained are analyzed. Finally, in Section 5, our conclusions and possible lines of future work are presented.

³ <https://www.top500.org/green500/>

2 Background and Related Works

First, the Intel Xeon Phi KNL and NVIDIA Pascal architectures are briefly described and compared. Then, the FW algorithm for all-pair shortest paths computation in a graph is described. Finally, some works related to this article are detailed.

2.1 Intel Xeon Phi KNL vs NVIDIA Pascal

Intel Xeon Phi KNL. Unlike a GPU or a co-processor such as KNC, KNL is a standalone processor, capable of booting operating systems and directly accessing DDR memory. The scalable unit of replication of the KNL architecture is the *tile*. Each tile houses 2 cores, a L2 cache shared between both cores, and a portion of the distributed directory. The cores of a tile implement *simultaneous multi-threading* (4 hw threads per core) and out-of-order execution in its pipeline, in addition to having 2 vector units that support AVX-512's new 512-bit instructions [14].

A KNL chip has between 32 and 36 active tiles (between 64 and 72 cores), depending on each specific model. The tiles are interconnected by a 2D mesh, with cache coherence based on distributed directory and MESIF protocol. This mesh can be configured in five different execution modes (cluster modes). Based on the execution mode selected, the distributed directory will be split among the tiles in the chip, which will have an impact on latency and bandwidth in memory access.

KNL comes with a 16GB HBM called MCDRAM, which is built into the same processor package. This memory can be configured in three different modes. In *flat* mode, the address space is mapped between the two memories (MCDRAM and DDR), so it is the programmer who has the responsibility of defining how to use them. On the other hand, *cache* mode leaves the system in charge of managing the MCDRAM as a DDR cache. Finally, the *hybrid* mode assigns part of the MCDRAM as flat, and part as cache [1].

NVIDIA Pascal. Pascal is the penultimate micro-architecture introduced by NVIDIA for the high-performance segment, successor to Maxwell. In this segment, a GPU of this family can have up to 3840 CUDA cores distributed among (at most) 60 Streaming Multiprocessors (SM). Compared to its predecessor, Pascal doubles the number of registers per core and increases the available size of shared memory.

This micro-architecture features several significant improvements over Maxwell. Among them, we can highlight the inclusion of an HBM of up to 16GiB in some of its chips, which allows them to reach a bandwidth of up to 720 GB/s. It also replaces the traditional PCIe bus used for CPU-GPU communication by a high-speed bus (NVLink), which significantly improves communication speed. Finally, the provision of a *unified memory*, consisting of a virtual address space between the CPU and GPU memories, aimed at simplifying programming [13].

Table 1. Intel Xeon Phi KNL 7230 vs NVIDIA Titan X

Device	Intel Xeon Phi KNL	NVIDIA Titan X
Chip	7230	GP102
Clock Frequency	1.3-1.5 GHz	1.42-1.53 GHz
Cores	64 (256 hw threads)	28 SMs (3584 CUDA cores)
Cache	1 MB L2	3 MB L2
SIMD	512-bit	-
HBM	16GB MCDRAM (450 GB/s)	-
RAM Memory	192GB DDR4 (115.2 GB/s)	12GB GDDR5X (480.4 GB/s)
Bus	-	PCI-Express 3.0 x16
Peak Theoretical performance SP (DP)	6 (3) TFLOPS	10.97 (0.342) TFLOPS
TDP	215W	250W
TFLOPS/W (SP/DP)	0.028 / 0.014	0.051 / 0.001
Launch Date	June 2016	August 2016

As regards peak performance, some Pascal chips can achieve double-precision (DP) throughput rates that are half of the single-precision (SP) ones. On the other hand, they can double SP performance if they apply half-precision computing [5].

Brief comparison. Table 1 presents a comparison of these architectures and, in particular, considers the models used in the experimental work. From the point of view of the theoretical peak performance in SP, the Titan X is far superior to the KNL 7230 (10.97 TFLOPS vs. 6 TFLOPS). However, due to the weak support of the former for DP, it is the KNL in this case that has a remarkable superiority (3 TFLOPS vs. 0.342 TFLOPS).

As regards main memory, the KNL has an HBM that puts it almost on par with Titan X in terms of bandwidth, since KNL has DDR4 technology while Titan X has GDDR5X. However, since it is a co-processor, the Titan X has a much smaller memory size than the KNL, which is a *host* unto itself.

Finally, when considering the (theoretical) energy efficiency, even though the Titan X has a higher thermal design power (TDP), its TFLOPS/W ratio in SP almost doubles that of KNL due to its higher theoretical performance peak. On the contrary, the poor performance of the Titan X for DP results in KNL being vastly superior in this case. Despite the fact that some years have passed since their launch, both architectures remain relevant, as shown by the latest edition of the Top500 ⁴ ranking, where 17 systems are equipped with Xeon Phi KNL and an additional, 30 with GPUs from the Pascal family.

2.2 All-Pair Shortest Paths Computation in a Graph

FW Algorithm. The pseudocode of FW is shown in Algorithm 2.2. Given a graph G of N vertexes, FW receives as input a dense $N \times N$ matrix D that

⁴ Top500 www.top500.org

Algorithm 1 Pseudocode of the FW algorithm

```
for  $k \leftarrow 0$  to  $N - 1$  do
  for  $i \leftarrow 0$  to  $N - 1$  do
    for  $j \leftarrow 0$  to  $N - 1$  do
      if  $D_{i,j} \geq D_{i,k} + D_{k,j}$  then
         $D_{i,j} \leftarrow D_{i,k} + D_{k,j}$ 
         $P_{i,j} \leftarrow k$ 
      end if
    end for
  end for
end for
```

contains the distances between all pairs of vertexes from G , where $D_{i,j}$ represents the distance from node i to node j ⁵. FW computes N iterations, evaluating in the k -th iteration all possible paths between vertexes i and j that have k as the intermediate vertex. As a result, it produces an updated matrix D , where $D_{i,j}$ now contains the shortest distance between nodes i and j up to that step. Also, FW builds an additional matrix P that records the paths associated with the shortest distances.

Blocked FW Algorithm. At first glance, the nested triple loop structure of this algorithm is similar to that of dense matrix multiplication (MM). However, since read and write operations are performed on the same matrix, the three loops cannot be freely exchanged, as is the case with MM. Despite this, the FW algorithm can be computed by blocks under certain conditions [21].

The blocked FW algorithm (BFW) divides matrix D into blocks of size $TB \times TB$, totaling $(N/TB)^2$ blocks. Computation is organized in $R = N/TB$ rounds, where each round consists of 4 phases ordered according to the data dependencies between the blocks:

1. Phase 1: Update the $D^{k,k}$ block because it only depends on itself.
2. Phase 2: Update the blocks in row k of blocks ($D^{k,*}$) because each of these depends on itself and on $D^{k,k}$.
3. Phase 3: Update the blocks in column k of blocks ($D^{*,k}$) because each of these depends on itself and on $D^{k,k}$.
4. Phase 4: Update the remaining $D^{i,j}$ blocks of the matrix because each of these depends on blocks $D^{i,k}$ and $D^{k,j}$ on its row and column of blocks, respectively.

Figure 1 shows each of the computation phases and the dependencies between blocks. The yellow squares represent blocks that are being computed, gray squares are those that have already been processed, and green squares are the ones that have not been computed yet. Last, arrows show the dependencies between blocks for each phase.

⁵ If there is no path between nodes i and j , their distance is considered to be infinite (usually represented as the largest positive value)

2.3 Related Works

The comparison of HPC architectures is a topic widely studied by the scientific community. In particular, there are several articles involving comparative studies between Xeon Phi KNL and NVIDIA Pascal in the field of large-scale economic modeling [19] linear algebra [4], computational fluid dynamics [15], and automatic deep learning [6], among others. However, as far as we know, this is the first to consider graph algorithms, in particular the FW algorithm for all-pair shortest paths.

The contributions of this work can be seen as an extension of a previous work of the authors [2], where the comparison of HPC architectures just considered performance and theoretical energy efficiency. By incorporating actual power consumption and programming cost to the comparison, this work is able to offer a more comprehensive analysis of the pro and cons of each architecture for graph applications.

3 FW Optimization

This section describes the Xeon Phi implementation followed by the GPU one.

3.1 Implementation on Xeon Phi KNL 7230

The implementation used considers the following optimizations:

- *Data locality.* By computing with BFW, it is not only possible to exploit data locality, but it is also possible to increase the parallelism available in the application.
- *Parallelism at thread level.* Using OpenMP, a multi-threaded version is obtained. Both Phase 2 and Phase 3 blocks are distributed among the different threads by means of the `for` directive with `dynamic` scheduling. In the case of Phase 1, since it consists of a single block, the iterations within it are distributed among the threads.

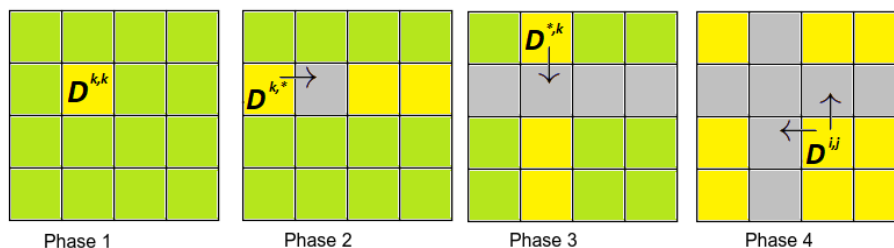


Fig. 1. BFW computation phases and block dependencies

- *Parallelism at data level.* Using the OpenMP `simd` directive, the operations of the innermost loop are vectorized when computing each block, which allows taking advantage of the AVX-512 instructions.
- *Loop unrolling.* By fully unrolling the innermost loop and loop i only once.
- *Branch prediction.* By including the built-in `_builtin_expect` compiler macro, `if` statement branches can be better predicted.
- *MCDRAM.* Since this is a bandwidth-limited application, using this special memory is greatly beneficial. Executions are done using the `numactl` command.

It should be noted that this implementation can be considered as an optimized version of [16], since it also includes intra-block parallelization for Phase 1 and the improvement in branch predictions.

3.2 Implementation on NVIDIA Titan X

The implementation used considers the optimizations known for GPU-based FW solutions at the moment [10,11]. Among those, the following can be mentioned:

- *Concurrency.* Three kernels have been developed that are invoked once for each BFW computation round. On round k , the first kernel computes block $D^{k,k}$ (Phase 1) and is instantiated with a single grid made up of a single block. Next, the second kernel is invoked, which computes both Phase 2 and Phase 3 blocks ($D^{k,*}$, $D^{*,k}$), and is instantiated with a single grid of $2 \times (R - 1)$ blocks. Finally, the third kernel, which computes the remaining blocks corresponding to Phase 4 ($D^{i,j}$), is invoked. This kernel is instantiated with a single grid of $(R - 1)^2$ blocks. In all cases, blocks are made up of $TB \times TB$ threads.
- *Exploitation of memory hierarchy.* For BFW computation, the use of shared memory is not only convenient but also necessary, especially in Phases 1-3 since the threads read and write to the same blocks of the matrix due to their dependencies. In the case of Phase 4, it is possible to take advantage of the private memory for the write block ($D^{i,j}$), which improves access times even more. Finally, the main memory accesses were organized so that they are coalescent.
- *Resource occupation.* In order to optimize this aspect, different thread block sizes were tried (using $TB = \{8, 16, 32\}$) to find the one that leads to the maximum possible number of active *warps*.
- *Loop unrolling.* By fully unrolling the loop that computes each thread.

4 Experimental Results

In this section, the experimental setup and methodology are described. Next, the results found are presented and analyzed. Last, the limitations of this research are mentioned.

4.1 Experimental Setup and Methodology

The tests were carried out on two different platforms ⁶. On the one hand, an Intel Xeon Phi KNL 7230 server configured in *all-to-all* cluster mode and with *flat* memory (ICC v19.0.0.117). On the other, an Intel Core i7-7700 3.6 GHz and 16GB RAM, which integrates an NVIDIA Titan X GPU (CUDA v9.0).

For both platforms, the variation in the size of the distances matrix ($N = \{4096, 8192, 16384, 32768, 65536\}$) and data type (*float*, *double*) were considered. In the case of KNL, different values for both the number of OpenMP threads $T = \{64, 128, 192, 256\}$ and $TB = \{16, 32, 64, 128\}$ were tried to identify the optimal values of $T_{float} = 128$, $T_{double} = 64$ and $TB = 64$. As regards the GPU, the best performances were obtained when using $TB_{float} = 32$ and $TB_{double} = 16$. Finally, to minimize variability, each specific test was repeated 15 times and the average values were calculated.

Since this paper considers power consumption as well as performance, the measurement environment used on each platform is described as follows:

- *Intel Xeon Phi KNL*. Intel has developed the Intel PCM ⁷ (Performance Counter Monitor) to take power measurements on both Intel Xeon and Xeon Phi processors. With Intel PCM interface, any programmer can perform an analysis of CPU resource consumption by means of hardware counters.
- *NVIDIA Titan X*. In modern NVIDIA GPUs, the NVIDIA System Management Interface utility (*nvidia-smi* ⁸) can be used to query power consumption at runtime. This tool is based on the NVIDIA Management Library (NVML) and is intended to help in the management and monitorization of NVIDIA GPU devices.

4.2 Performance Results

The GFLOPS metric is used for performance evaluation:

$$GFLOPS = \frac{2 \times N^3}{t \times 10^9} \quad (1)$$

where N is the size of the distances matrix, t is execution time (in seconds), and factor 2 represents the number of floating-point operations required by each iteration of the innermost loop.

Figure 2 shows the performance obtained by each implementation with different values for both matrix and data type used. In SP (*float*), it can be seen that the GPU achieves a better performance with smaller matrix sizes, being approximately 19% higher when $N = 4096$. In these cases, GPU is better suited than KNL, which requires higher workloads to reach its maximum use. This is reflected in the graph, since, as the size of the distances matrix increases, KNL

⁶ The characteristics of each platform were described at the end of Section 2.1

⁷ Intel Performance Counter Monitor: <http://www.intel.com/software/pcm>

⁸ NVIDIA System Management Interface: <https://developer.nvidia.com/nvidia-system-management-interface>

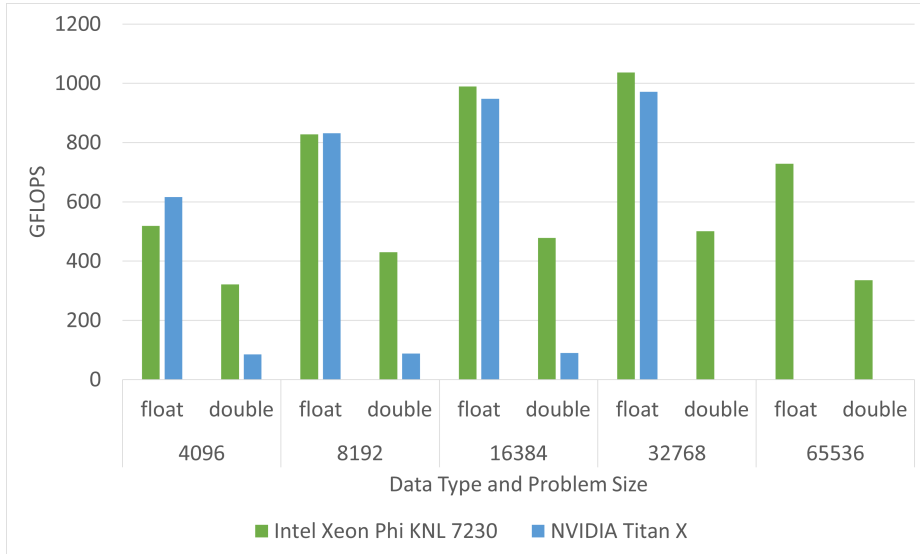


Fig. 2. Performance obtained with different data types and distances matrix sizes with Intel Xeon Phi KNL 7230 and NVIDIA Titan X

achieves the best performance, reaching an additional 7% difference. It should be noted that with $N = 65536$, KNL experiences a performance loss of approximately 30%, due to the fact that, with this value, the available size of the MCDRAM is exceeded and partial use of DDR4 is required, which has a much lower bandwidth. Even so, it is more convenient than using GPU, which cannot compute cases with $N > 32768$ because the available main memory space is exceeded⁹.

As regards DP (*double*), the results obtained are as expected due to the weak support of Titan X for this class of operations. While GPU obtains an almost constant performance of ~ 90 GFLOPS, KNL improves its performance as N increases and eventually surpasses the size of the MCDRAM. In particular, KNL gets about half the FLOPS of SP but improves to $4.3\times$ those of Titan X. Lastly, the memory limit issue in GPU is also worse due to the fact that the *double* type requires more space (cases with $N > 16384$ could not be processed).

4.3 Power and Energy Efficiency Results

As mentioned in Section 1, application performance is not the only point of interest to be considered, energy efficiency also matters. Table 2 lists energy efficiency ratios taking into account the GFLOPS peaks reached and the TDP

⁹ Naturally, it is also possible to develop an implementation that processes the matrix in parts and does not have this memory limitation. However, the need to run I/O operations for each round would significantly degrade performance.

Table 2. Theoretical comparison of performance and power efficiency by platform

Precision	Platform	GFLOPS (peak)	TDP (Watt)	GFLOPS/Watt
SP	<i>Xeon Phi KNL 7230</i>	1037	215	4.82
	<i>NVIDIA Titan X</i>	972	250	3.89
DP	<i>Xeon Phi KNL 7230</i>	501	215	2.33
	<i>NVIDIA Titan X</i>	90	250	0.36

Table 3. Comparison of performance and power efficiency by platform

Precision	Platform	GFLOPS (peak)	Power (Watt)	GFLOPS/Watt
SP	<i>Xeon Phi KNL 7230</i>	1037	229.7	4.51
	<i>NVIDIA Titan X</i>	972	209.5	4.64
DP	<i>Xeon Phi KNL 7230</i>	501	261	1.92
	<i>NVIDIA Titan X</i>	90	144.8	0.62

of the platforms used. As it can be seen, KNL is superior in both cases. In particular, KNL gets a GFLOPS/Watt ratio that is $1.2\times$ better than that of Titan X in SP. However, this factor increases up to $5.5\times$ in DP, due to the weak support of this GPU for these operations. It should be noted that for this analysis, *host* consumption by GPU was not taken into account, so the differences could be even greater.

TDP can be useful for qualitative comparison purposes and sometimes is the only valid metric when power measuring is not possible. However, actual power readings can differ from TDP due to a variety of reasons (power saving techniques available in modern processors/accelerators, specific workload in the target application, among others) [9]. For this reason, this work also includes an empirical power-performance analysis between platforms.

Table 3 shows a summary of the best performance and the corresponding average power consumption on the different architectures under study.

These power measurements reinforce the idea that actual power readings can differ from the TDP of each platform. In the KNL architecture, a higher power consumption is observed in both SP and DP scenarios; however, the difference is larger in the DP case. Hence, DP not only affects execution time but also increases power consumption, as it was also observed in other recent studies [8,18].

In opposite sense to the KNL case, average power consumption is lower than the corresponding TDP in Titan X. Considering this issue, Titan X becomes slightly better than KNL when SP is used. Nevertheless, it must be remarked that host power consumption is not included in the GPU estimation; so, KNL still remains as probably the best option from this perspective. Despite the fact that energy efficiency gets almost doubled in DP, this improvement is virtually useless due to the poor performance of the Titan X.

4.4 Programming Cost

As well as general-purpose architectures, KNL supports widely extended parallel programming models in HPC (such as OpenMP or MPI). On the other hand, CUDA is the *de facto* standard for GPU programming nowadays. This fact puts KNL in a favourable position with respect to GPUs, since code development and portability get simplified.

Beyond specific language learning, GPUs may require additional programming efforts. Even though it experienced a significant performance loss when matrix size exceeded that of the MCDRAM, KNL was able to process all graphs. On the other hand, Titan X was not able to process those that exceeded the size of its RAM memory. While it is possible to develop an implementation that does, it would also have an associated performance loss and would require additional programming effort (not required in the case of KNL).

4.5 Limitations

Two possible limitations of this research can be mentioned:

- In 2019, Intel cancelled the KNL line [20]. Even though, several features of these processors (like AVX-512 floating point unit, the mesh interconnect on the die, and the integration of high bandwidth stacked memory into the processor) have gone mainstream in the current Xeons [12]. Thus, part of the results found in this research can be extrapolated to Xeon processors.
- This analysis focuses on two specific architecture models and that, as such, some of the results found could change if different models were used. For example, the NVIDIA GP100 (Pascal) GPU has a slightly lower peak SP performance than the Titan X (10.1 TFLOPS). However, its support for DP is vastly higher, being half SP (5 TFLOPS). In this sense, it is considered that the comparison is equally valuable to show trends and the distinctive characteristics of each architecture, even when there are different models that may present variations in their specifications.

5 Conclusions and Future Work

This work focuses on the comparison of Intel Xeon Phi KNL and NVIDIA Pascal architectures. Taking the FW algorithm for computing all-pairs shortest paths in a graph as a case study, optimized implementations were used to compare the achievable performance on each platform and thus be able to extract some general guidelines. Among those, the following can be mentioned:

- Despite the fact that the preliminary analysis indicated that Titan X was far superior to KNL, the performances (SP) obtained for FW were comparable. While the GPU performed better for small graphs, as the size of the distances matrix increased, it was the KNL that performed better. This fact leads to KNL's need for large workloads in order to make the most of it.

- As regards energy efficiency, contrary to what was found in the preliminary analysis, no significant difference was observed in SP. This result contributes to the fact that device sustainable performance and energy efficiency vary depending on each particular problem and its corresponding software implementation.
- Beyond specific language learning, GPUs may require additional programming efforts due to their co-processor nature (they are not hosts per se) and limited memory sizes.

Future works include:

- Extending the GPU implementation to support graphs larger than the main memory size.
- Including other models of the studied architectures (especially Pascal GPUs).

The development of these activities would give greater robustness and representativeness to the study carried out.

Acknowledgments. The authors are grateful for the support of NVIDIA through the donation of the Titan X GPU used in this research.

References

1. Codreanu, V., Rodríguez, J., Saastad, O.W.: Best Practice Guide - Knights Landing (2017), <https://bit.ly/2CEo1bR>
2. Costanzo, M., Rucci, E., Costi, U., Chichizola, F., Naiouf, M.: Comparación de Arquitecturas HPC para Computar Caminos Mínimos en Grafos. Intel Xeon Phi KNL vs NVIDIA Pascal. In: Actas del XXVI Congreso Argentino de Ciencias de la Computación (CACIC 2020). pp. 82–92 (2020)
3. Deng, L., Bai, H., Zhao, D., Wang, F.: Kepler gpu vs. xeon phi: Performance case study with a high-order cfd application. In: 2015 IEEE International Conference on Computer and Communications (ICCC). pp. 87–94 (2015)
4. Deveci, M., Trott, C., Rajamanickam, S.: Multithreaded sparse matrix-matrix multiplication for many-core and gpu architectures. *Parallel Computing* **78**, 33 – 46 (2018). <https://doi.org/https://doi.org/10.1016/j.parco.2018.06.009>, <http://www.sciencedirect.com/science/article/pii/S0167819118301923>
5. Foley, D., Danskin, J.: Ultra-performance pascal gpu and nvlinc interconnect. *IEEE Micro* **37**(2), 7–17 (2017)
6. Gawande, N.A., Daily, J.A., Siegel, C., Tallent, N.R., Vishnu, A.: Scaling deep learning workloads: Nvidia dgx-1/pascal and intel knights landing. *Future Generation Computer Systems* **108**, 1162 – 1172 (2020)
7. Giefers, H., Staar, P., Bekas, C., Hagleitner, C.: Analyzing the energy-efficiency of sparse matrix multiplication on heterogeneous systems: A comparative study of gpu, xeon phi and fpga. In: 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). pp. 46–56 (2016)
8. Hashemi, S., Anthony, N., Tann, H., Bahar, R.I., Reda, S.: Understanding the impact of precision quantization on the accuracy and energy of neural networks. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2017. pp. 1474–1479 (2017). <https://doi.org/10.23919/DATE.2017.7927224>

9. Igual, F.D., García, C., Botella, G., Piñuel, L., Prieto-Matías, M., Tirado, F.: Non-negative matrix factorization on low-power architectures and accelerators. *Comput. Electr. Eng.* **46**(C), 139–156 (Aug 2015). <https://doi.org/10.1016/j.compeleceng.2015.03.035>, <https://doi.org/10.1016/j.compeleceng.2015.03.035>
10. Katz, G.J., Kider, Jr, J.T.: All-pairs shortest-paths for large graphs on the gpu. In: Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware. pp. 47–55. GH '08, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2008)
11. Lund, B.D., Smith, J.W.: A multi-stage CUDA kernel for floyd-warshall. *CoRR* **abs/1001.4108** (2010), <http://arxiv.org/abs/1001.4108>
12. Morgan, T.P.: The end of Xeon Phi - It's Xeon and Maybe GPUs from here (2018), <https://www.green500.org/>
13. NVIDIA: NVIDIA Tesla P100., <https://bit.ly/20zrrk1>
14. Reinders, J., Jeffers, J., Sodani, A.: Intel Xeon Phi Processor High Performance Programming Knights Landing Edition. Morgan Kaufmann Publishers Inc., Boston, MA, USA (2016)
15. Robertsén, F., Mattila, K., Westerholm, J.: High-performance SIMD implementation of the lattice-Boltzmann method on the Xeon Phi processor. *Concurrency and Computation: Practice and Experience* **31**(13) (7 2019). <https://doi.org/10.1002/cpe.5072>
16. Rucci, E., De Giusti, A., Naiouf, M.: Blocked All-Pairs Shortest Paths Algorithm on Intel Xeon Phi KNL Processor: A Case Study. In: De Giusti, A.E. (ed.) *Computer Science – CACIC 2017*. pp. 47–57. Springer Int. Pub., Cham (2018)
17. Rucci, E., Garcia, C., Botella, G., De Giusti, A., Naiouf, M., Prieto-Matias, M.: SWIFOLD: Smith-Waterman implementation on FPGA with OpenCL for long DNA sequences. *BMC Systems Biology* **12**(5), 96 (Nov 2018). <https://doi.org/10.1186/s12918-018-0614-6>
18. Sakamoto, R., Kondo, M., Fujita, K., Ichimura, T., Nakajima, K.: The effectiveness of low-precision floating arithmetic on numerical codes: A case study on power consumption. In: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region. p. 199–206. HPCA-sia2020, Association for Computing Machinery, New York, NY, USA (2020). <https://doi.org/10.1145/3368474.3368492>, <https://doi.org/10.1145/3368474.3368492>
19. Scheidegger, S., Mikushin, D., Kubler, F., Schenk, O.: Rethinking large-scale economic modeling for efficiency: Optimizations for gpu and xeon phi clusters. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 610–619 (2018)
20. Trader, T.: Requiem for a Phi: Knights Landing Discontinued (2018), <https://www.hpcwire.com/2018/07/25/end-of-the-road-for-knights-landing-phi>
21. Venkataraman, G., Sahni, S., Mukhopadhyaya, S.: A Blocked All-Pairs Shortest-Paths Algorithm, pp. 419–432. Springer Berlin Heidelberg (2000). https://doi.org/10.1007/3-540-44985-X_36
22. Véstias, M., Neto, H.: Trends of cpu, gpu and fpga for high-performance computing. In: 2014 24th International Conference on Field Programmable Logic and Applications (FPL). pp. 1–6 (Sept 2014). <https://doi.org/10.1109/FPL.2014.6927483>