Work Distribution of Data-Parallel Applications on Heterogeneous Systems

Suejb Memeti(✉) and Sabri Pllana

Department of Computer Science, Linnaeus University, 351 95 Vaxjo, Sweden
{suejb.memeti,sabri.pllana}@lnu.se

Abstract. Heterogeneous computing systems offer high peak performance and energy efficiency, and utilizing this potential is essential to achieve extreme-scale performance. However, optimal sharing of the work among processing elements in heterogeneous systems is not straightforward. In this paper, we propose an approach that uses combinatorial optimization to search for optimal system configuration in a given parameter space. The optimization goal is to determine the number of threads, thread affinities, and workload partitioning, such that the overall execution time is minimized. For combinatorial optimization we use the Simulated Annealing. We evaluate our approach with a DNA sequence analysis application on a heterogeneous platform that comprises two Intel Xeon E5 processors and an Intel Xeon Phi 7120P co-processor. The obtained results demonstrate that using the near-optimal system configuration, determined by our algorithm based on the simulated annealing, application performance is improved.

1 Introduction

Heterogeneous computing systems consist of general-purpose CPUs and accelerators – such as, graphical processing units (GPUs) or Intel Xeon Phi – which offer high performance and energy efficiency. Some of the most powerful supercomputers in the TOP500 [1] list are heterogeneous at their node level. For example, Tianhe-2 nodes consist of two Intel IvyBridge CPUs and three Intel Xeon Phi co-processors, whereas a node of Titan contains one AMD Opteron CPU and one Nvidia Tesla GPU. Mapping computations to processing elements of the heterogeneous node in an optimal way is an important step to efficiently utilize the large-scale computing systems [2,24].

Due to the different performance characteristics of heterogeneous processing elements, distributing the workload across such elements to utilize the aggregate power of heterogeneous systems depends on many parameters and is a non-trivial task. Using enumeration of all possible parameters to determine the optimal system configuration is prohibitively time-consuming. Equation 1 shows the product function of the parameter value ranges that determines the number

This research has received funding from the Swedish Knowledge Foundation under Grant No. 20150088.

M. Taufer et al. (Eds.): ISC High Performance Workshops 2016, LNCS 9945, pp. 69–81, 2016.
DOI: 10.1007/978-3-319-46079-6_6

of all possible configurations, where $C = \{c_1, c_2, \ldots, c_m\}$ is a set of parameters and each parameter c_i has a value range R_{c_i}.

$$\prod_{i=1}^{m} R_{c_i} = R_{c_1} \times R_{c_2} \times \ldots \times R_{c_m} \quad (1)$$

Various techniques have been proposed for utilization of heterogeneous computing systems. CoreTSAR [27] is an adaptive work-sharing library for scheduling computations across multiple devices. Qilin [17] is an off-line based profiling technique for automatic mapping of computations to processing elements. Grewe and O'Boyle [11] use a static partitioning approach based on machine learning methods to distribute OpenCL programs on heterogeneous computing systems. A task splitting and distribution dynamic scheduling technique was proposed by Ravi and Agrawal [26]. Albayrak et al. [3] use the Greedy Algorithm to determine the near-optimal mapping for applications designed as sequence of kernels.

So far not much research has addressed combinatorial optimization approaches for workload distribution across resources of heterogeneous systems. Furthermore, related research focuses on heterogeneous systems that are accelerated with GPUs. Platforms accelerated with the Intel Xeon Phi deserve our attention because of their capability to deliver high performance, energy efficiency, and the ease of programmability and portability [6,8,12].

In this paper we propose an optimization approach that uses combinatorial optimization to determine near-optimal system configuration parameters (including the number of threads, thread affinity, and the workload distribution ratio) of a heterogeneous systems. To search for the optimal system configuration in the given large discrete search space we use Simulated Annealing [25]. The optimization goal is to minimize the application's execution time. For empirical evaluation we use a parallel application for DNA sequence analysis of real world DNA sequences of various animals. We perform our experiments on a heterogeneous platform that comprises two Intel Xeon E5 CPUs and an Intel Xeon Phi7120P co-processor.

Results demonstrate that by running only about 5 % of all the possible experiments we can determine a near-optimal system configuration, which yields with 1.74× speedup and 2.2× compared to the case when only the cores of host or device are used.

The major contributions of this paper include:

1. a heuristic based optimization approach to determine the near-optimal system configuration (such as, workload distribution ratio between host and device, number of threads and thread allocations),
2. a parallel algorithm for matching patterns in DNA sequences that efficiently utilizes the resources of the host and device in heterogeneous systems,
3. an experimental evaluation of our approach for DNA sequence analysis using real-world DNA sequences.

The rest of the paper is organized as follows. Section 2 provides background information with respect to meta-heuristics and heterogeneous computing systems. In Sect. 3.2 we describe the methodology, including the heuristic-guided

approach for optimization of workload distribution in heterogeneous systems, and our algorithm for DNA sequence analysis. Section 4 presents the experimentation environment and discusses the experimental evaluation results. The related work will be discussed in Sect. 5. Section 6 provides a conclusion and discusses the future work.

2 Background

In this section we provide background information on the meta-heuristics and a heterogeneous computing platform that is accelerated with the Intel Xeon Phi co-processor.

2.1 Meta Heuristics

Meta-heuristics are designed for finding, generating or selecting the global optimum on some class of problems with less computation effort, which in general is a very difficult problem. A well known problem that can be solved using meta-heuristics is the Traveling Salesman Problem (TSP) where the search-space grows exponentially as the problem size increases. Brute-force (or enumeration) approaches are infeasible to deal with such complex problems.

As there are many different heuristic-based optimization methods, such as Genetic Algorithms, Ant Colony Optimization, Simulated Annealing, Local Search, and Tabu Search, which differ substantially in their underlying concepts, choosing the most convenient requires to consider different characteristics [7]. Such characteristics include: generation of new solutions, treatment of the new solutions, number of search agents, limitations of the search space, prior knowledge, flexibility for specific constraints, ease of implementation, computational complexity, convergence speed, reliability, type of optimization problem and search space, the available computational time, or the demanded solution quality [25].

2.2 Heterogeneous Systems – Intel Xeon Phi

A typical heterogeneous node that uses the Intel Xeon Phi as accelerator may consist of one or two CPUs on the host, and one to eight accelerators. Our *Emil* system that is used for experimentation in this paper comprises two Intel Xeon E5 2695 v2 and one Intel Xeon Phi 7120P co-processor. The E5 CPUs comprise 12 Ivy Bridge cores. These cores are connected using a ring topology, which features low latency and high throughput. The L3 cache size is 30 MB. The CPUs are connected to the memory using the Quad channel memory controller. The two host CPUs are linked through the QuickPath Interconnect, which offers up to 8.0 GT/s.

The Intel Xeon Phi is a many-core share memory processor. The lightweights Linux Operating System running on the card enables communicating with it over ssh. In the current version (Knights Landing, used in this paper) there are

61 cores, each of them has four hardware threads. The base core's frequency is 1.2 GHz, and 1.3 GHz max turbo frequency [8]. A 30.5 MB unified L2 cache is formed through a bidirectional ring bus interconnect that connects these cores. The 16 memory channels offer a theoretical maximum memory bandwidth of 352 GB/s.

The Intel Xeon Phi supports 512-bit wide Single Instruction Multiple Data (SIMD) registers that can perform 16 single precision floating point operations, or eight double precision floating point operations per cycle. The theoretical single and double performance capability of the Intel Xeon Phi is one and two teraFLOP/s, respectively. The practical performance capabilities of the Intel Xeon Phi, and its accessibility from the programmability point of view have been investigated in different articles [9,16,28].

3 Methodology

This section describes our heuristic-guided approach for optimization of workload distribution on heterogeneous computing systems. Furthermore, it describes our parallel algorithm for DNA sequence analysis that is able to utilize the aggregate power of host CPUs and accelerators in heterogeneous computing systems.

3.1 Using Simulated Annealing for Optimization of Heterogeneous Systems

Simulated Annealing (SA) is an optimization technique used to approximate global optimization in large discrete search space. A fundamental property of SA is its ability to accept worse solutions that allows a more extensive search of the optimal space.

The method and its name is inspired by the process of material cooling and annealing, where the slow cooling is interpreted as a slow decrease in the probability of accepting worse solutions.

While the temperature T is higher, it is more likely to accept new solutions. Therefore, there is a corresponding chance to get out of a local minimum, in favor of searching for a global optimum. The lower the temperature, less likely it accepts new solutions [25].

In the context of optimizing the workload distribution on heterogeneous systems, the configuration space is as follows:

- workload fraction is a discrete value from 0–100
- number of threads used for the host (1–48) and device (1–244);
- thread allocation strategy for the host (none, compact, scatter) and device (balanced, compact, scatter).

The objective function E (analog of *energy*) that we are trying to minimize is the total *execution time* of the application running on the host and the device, which basically is determined by the maximum of the t_{host} and t_{device}:

$$E = max(t_{host}, t_{device}) \tag{2}$$

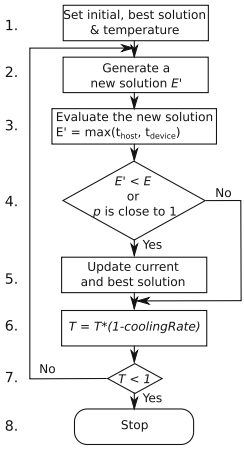


Fig. 1. The major steps of the simulated annealing based algorithm.

Figure 1 shows the major steps of the simulated annealing algorithm. First, we set the initial value of T (temperature) and generate a random initial solution (step 1). Thereafter, we generate a new solution (step 2), and evaluate it (step 3). If the newly generated solution is better than the current one, or the probability distribution is close to 1 (step 4) we update the current and best solution (step 5), otherwise we decrease the temperature (step 6). Unless the temperature has cooled down, the steps 2–6 will be repeated.

The annealing schedule *coolingRate* is defined as follows:

$$T = T * (1 - coolingRate); \tag{3}$$

Equation 4 shows the Boltzmann probability distribution [25] (acceptance function) used to decide whether or not to accept a worse solution. If the energy of the newly generated solution E' is lower than the energy of the current solution E, then we accept it unconditionally, otherwise we consider temperature and the time difference between the two solutions being compared. The higher the temperature, it is more likely that the system accepts worse solutions.

$$p = exp((E - E')/T) \tag{4}$$

3.2 DNA Sequence Analysis on Heterogeneous Computing Nodes

The current version of Intel Xeon Phi co-processor (Knights Corner) offers two programming models:

- *offload* - where parts of the code are offloaded to the co-processor
- *native* - where the code is compiled specifically for running natively on the co-processor. The code and dependent libraries are transferred to the device.

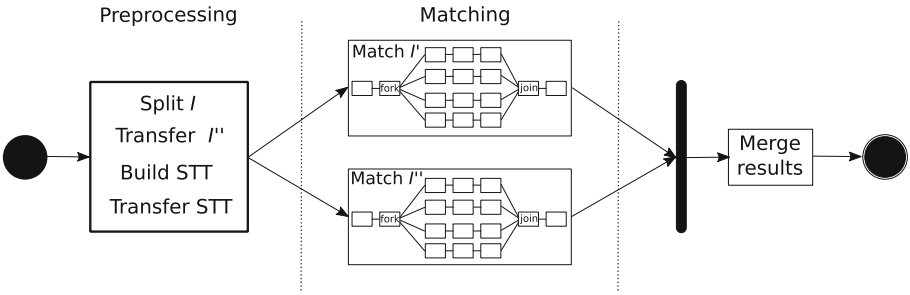


Fig. 2. Using resources of the host and device for DNA sequence analysis.

Our approach for parallel DNA Sequence analysis is based on the *offload* programming model, which allows using the resources of the host and the device at the same time. Figure 2 depicts workload distribution (that is partitioning) strategy of the DNA Sequences to be analyzed in heterogeneous systems. In the preprocessing phase the input DNA sequence I is split based on the fraction ratio F (selected by SA) into the part $I' = (F/100) * I$ that will be processed on host CPUs, and $I'' = I - I'$ processed on device. During this phase the construction of the State Transition Table (STT) takes place. When ready, both I'' and the STT are transferred to the device.

Our application takes advantage of the double advantage of transforming-and-tunning [8] of the Intel Xeon Phi, which means that with not much investment we can use the same algorithm for matching patterns in both host and device. When the process of matching I'' on the device is completed, we transfers the result (the total number and the location of matched patterns) to the host memory, and a merge of the results is performed.

4 Evaluation

In this section we empirically evaluate our heuristic-guided approach for optimization of DNA sequence analysis on heterogeneous platforms. We describe the following,

- the experimentation environment,
- performance comparison of our heuristic-guided approach with the enumeration approach (also known as brute force),
- the performance improvement when using the selected solution by our approach (that uses both resources of host and device) compared to host-only (48-threads) and device-only (244-threads) executions.

4.1 Experimentation Environment

In this section we provide information related to the experimentation environment including: (1) system configuration, (2) benchmark application, (3) data

sets used for evaluation of our approach, and (4) parameter values that define the system configuration space.

System Configuration. The heterogeneous system used for the performed experiments consists of two Intel Xeon E5 processors and one Intel Xeon Phi 7120P co-processor.

Benchmark Application. We used a DNA sequence analysis application with real-world DNA sequences. The major features of our *Emil* system at the Linnaeus University and implementation details of our algorithm for DNA sequence analysis are described in [19,20].

Data Sets. Real world DNA sequences of human (3.17 GB), mouse (2.77 GB), cat (2.43 GB), and dog (2.38 GB) extracted from the GenBank sequence database of the National Center for Biological Information [22]. We use patterns from the *regex-dna* benchmark for matching and extracting specific k-mers from a DNA sequence. The PaREM [18] tool is used to generate the STT for the used patterns.

System Configuration Space. The parameters and their value ranges that define the system configuration for our combinatorial optimization approach are shown in Table 1.

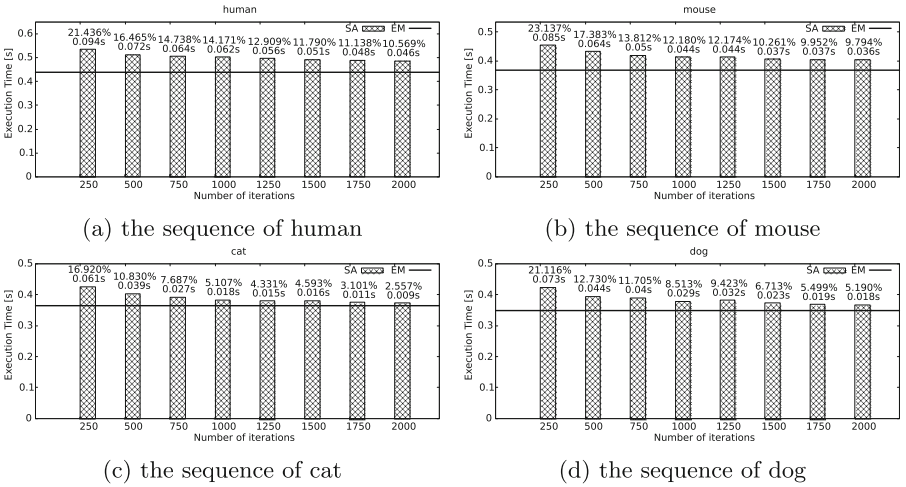
Table 1. The parameters that define the system configuration

Parameter Name	Type	Value range	
		Host	Device
Number of threads	Discrete	{2, 6, 12, 24, 36, 48}	{2, 4, 8, 16, 30, 60, 120, 180, 240}
Thread affinity	Discrete	{none, compact, scatter}	{balanced, compact, scatter}
DNA sequence fraction	Discrete	{0,...,100}	100 - (Host sequence fraction)

4.2 Performance Comparison of Our Heuristic-Guided Optimization Approach with Enumeration

The enumeration approach certainly determines the optimal system parameter values, which results with the best performance by trying all the possible parameter values. However, for large search space of real-world problems, this approach is prohibitively time consuming. For example, despite the fact that we tested only what we considered reasonable parameter values (see Table 1), a total of 19926 experiments were required by enumeration to determine the optimal system configuration. We have achieved comparatively good performance results by using our heuristic-guided approach based on Simulated Annealing by trying only a relatively small subset of the total experiments involved in enumeration.

For performance comparison, we use the absolute difference $|t_{EM} - t_{SA}|$ and the percent difference $100 \cdot absolute\_difference/t_{EM}$, where t_{EM} indicates the best execution time determined using enumeration, and t_{SA} indicates the execution time of our algorithm with a system configuration suggested by the simulated annealing approach.



(a) the sequence of human

(b) the sequence of mouse

(c) the sequence of cat

(d) the sequence of dog

Fig. 3. Performance comparison between the best system configuration determined by the enumeration [EM] approach and the near to optimal one determined by the Simulated Annealing [SA]. The labels at the top of each bar depicts the percent difference [%] and absolute difference [s].

The results in Fig. 3 show the execution time of our algorithm when using the system configuration suggested by SA for various iterations (experiments performed by SA) compared to the best performance achieved using the system configuration determined by enumeration. The labels on top of each bar indicate the percent difference and absolute difference. The horizontal line indicates the execution time of the system configuration determined with enumeration.

By running about 1000 experiments (that is $100 \times 1000/19926 = 5\,\%$ of the total experiments required by enumeration) SA suggests system configuration that yields with a performance that is close to the optimal one determined by enumeration. Please note that SA is a global optimization approach and to avoid ending at a local optima during the space exploration, sometimes it accepts worse system configuration that results with a higher execution time compared to previous iterations.

The percent difference is shown in the labels on top of the bars (row 1) of Fig. 3. While the average percent difference of SA with 250 iterations is high (20.6 %) compared to enumeration, it decreases significantly by increasing the number of iterations. For example by increasing the number of iterations to 500, 750 and 1000 the percent difference decreases to 14.3 %, 11.9 % and 9.9 % respectively.

The second row of the labels on top of the bars of Fig. 3 shows the absolute difference of SA compared to enumeration. The average absolute difference for 250 iterations is 0.078 s, whereas for 500, 750 and 1000 iterations it is only 0.055, 0.045, and 0.038 s, respectively.

4.3 Performance Improvement

This section presents the performance improvement achieved in case all available resources of the host and device are used for the DNA sequence analysis compared to host- and device-only.

The results in Fig. 4 expose the accomplished performance improvement (speedup) when the system configuration determined by the simulated annealing algorithm or the enumeration approach is used for DNA sequence analysis compared only to the host. We may observe that as we increase the number of iterations the speedup of simulated annealing approaches the maximal speedup determined by enumeration. The average speedup achieved for 250, 500, and 1000

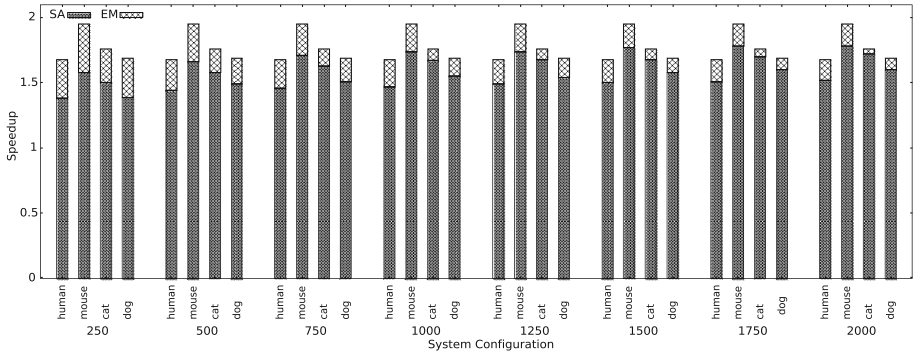


Fig. 4. Speedup achieved when host and device are used for the DNA sequence analysis compared with the host only. We consider 19926 system configurations determined by enumeration (EM) and by Simulated Annealing after 250, 500, 750, 1000, 1250, 1500, 1750, 2000 iterations.

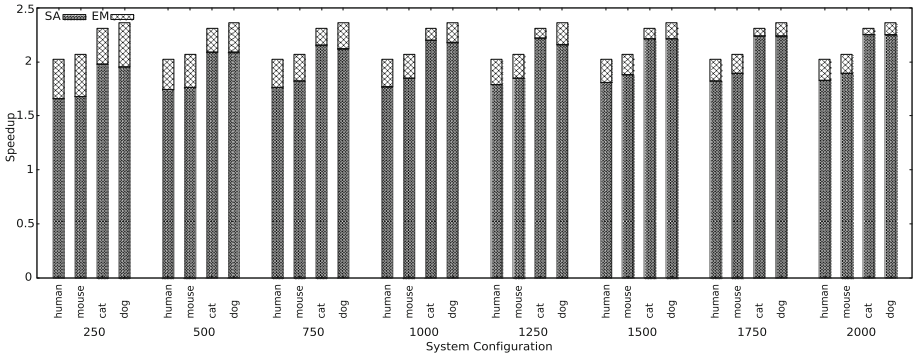


Fig. 5. Speedup achieved when host and device are used for the DNA sequence analysis compared with the device only. We consider system configurations determined by enumeration (EM) and by Simulated Annealing after 250, 500, 750, 1000, 1250, 1500, 1750, 2000 iterations.eps

iterations is 1.46×, 1.54, and 1.63× whereas the maximal achieved speedup with enumeration is 1.77×.

Figure 5 depicts the achieved speedup when the system configuration determined by the simulated annealing algorithm or the enumeration approach is used for the DNA sequence analysis compared only to the device. The average speedup achieved for 250, 500, and 1000 iterations is 1.82×, 1.92, and 2.0× whereas the maximal achieved speedup with enumeration is 2.2×.

5 Related Work

Utilizing the combined computation power of multi-core CPUs and many-core accelerators in heterogeneous systems is important to achieve high performance. Various approaches to distribute the workload across different devices in heterogeneous systems have been proposed.

Scogland et al. [27] proposed an adaptive worksharing library to schedule computational load across devices. Their extension of accelerated OpenMP evaluates the speed of each device statically, then use these indicators to automatically split the workload across different devices.

Similarly Ayguadé et al. [5] investigated the extension of OpenMP to allow workload distribution on future iterations based on the results of first static ones.

While Scogland et al. [27] and Ayguadé et al. [5] tend to offer solutions that require minimal changes to the original source code, task block models, such as StarPU [4] and OmpSs [10] require the user to determine workload distribution manually and may require significant structural changes to the original serial code.

Odajima et al. [23] proposed an approach that combines the pragma-based XcalableMP (XMP) [21] programming language with the runtime system by StarPU to utilize both GPU and CPU resources on each node for work distribution of the loop executions. They use the XMP for data distribution and synchronization purposes, whereas the StarPU is used for scheduling the tasks among host CPUs and accelerating devices.

Qilin [17] is a programming system that is based on a regression model to predict the execution time of kernels. It uses off-line learning that is thereafter used in compile time to predict the execution time for different input sizes and system configurations.

Ravi and Agrawal [26] proposed their dynamic scheduling framework that divides tasks into smaller ones that later on are distributed across different processing elements in a task-farm way.

Dokulili et al. [9] proposed a C++ framework for dynamic distribution of the work among the host CPUs and co-processor devices. The workload is distributed using a priority queue technique, where one core of the host is responsible for the queue management.

Grewe and O'Boyle [11] proposed a static partitioning approach to distribute OpenCL programs on heterogeneous systems. Their approach is based on static analysis to extract code features from OpenCL programs. These features are

then used to determine the best partitioning across the different devices. Their approach relies on the architectural characteristics of a system.

In comparison to the aforementioned approaches, we use combinatorial optimization to determine the near-optimal system configuration.

Albayrak et al. [3] propose a profiling-based approach for mapping kernel computations to heterogeneous platforms. Their approach extracts profiling information by running each application on each device (including host CPUs and accelerators), to collect information such as execution time and data transfer time. This information is then passed to solvers such as Greedy Algorithm to select the optimal mapping for a specific kernel.

In contrast to Albayrak et al. [3] we use Simulated Annealing to minimize the overall execution cost. Furthermore, they work focuses on applications that are designed as sequence of kernels, whereas we target data-parallel applications.

In the context of Grid computing environments, Kołodziej et al. [15] have studied the use of meta-heuristics for efficient data scheduling. Khan et al. [14] address *parameter sensitivity* [13] of workflows and propose to use the Ant-Colony Optimization to identify parameters of workflow activities that affect more the overall result of the workflow.

6 Conclusion and Future Work

In this paper we have presented a combinatorial optimization approach to determine the system configuration (the number of threads, thread affinity, and the DNA sequence fraction for the host and device) such that the overall execution time is minimized. Furthermore, we presented an approach for DNA sequence analysis that is designed to efficiently utilize the available resources of heterogeneous systems accelerated with Intel Xeon Phi.

Determining the best system configuration using enumeration is prohibitively time consuming because it requires many experiments. Using our approach we were able to determine a near optimal system configuration by executing only about 5 % of experiments, which results with comparable performance to the one determined with enumeration. When using the near-optimal system configuration selected by our approach we achieved a maximal speedup of 1.74× compared to host-only execution, and up to 2.2× speedup compared to device-only execution.

Future work will study multi-objective optimization (energy consumption and performance efficiency) of DNA sequence analysis using platforms accelerated with the second generation of the Intel Xeon Phi (Knights Landing).

References

1. TOP500 Supercomputer Sites. http://www.top500.org/. Accessed Jan 2016
2. Abraham, E., Bekas, C., Brandic, I., Genaim, S., Johnsen, E.B.,Kondov, I., Pllana, S., Streit, A.: Preparing HPC applications for exascale: challenges and recommendations. In: 2015 18th International Conference on Network-Based Information Systems (NBiS), pp. 401–406, September 2015

3. Albayrak, O.E., Akturk, I., Ozturk, O.: Improving application behavior on heterogeneous manycore systems through kernel mapping. Parallel Comput. **39**(12), 867–878 (2013). http://dx.doi.org/10.1016/j.parco.2013.08.011
4. Augonnet, C., Thibault, S., Namyst, R., Wacrenier, P.A.: StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. Concurr. Comput. Pract. Exp. **23**(2), 187–198 (2011)
5. Ayguadé, E., Blainey, B., Duran, A., Labarta, J., Martínez, F., Martorell, X., Silvera, R.: Is the *Schedule* clause really necessary in OpenMP? In: Voss, M.J. (ed.) WOMPAT 2003. LNCS, vol. 2716, pp. 147–159. Springer, Heidelberg (2003). doi:10.1007/3-540-45009-2_12
6. Benkner, S., Pllana, S., Traff, J., Tsigas, P., Dolinsky, U., Augonnet, C., Bachmayer, B., Kessler, C., Moloney, D., Osipov, V.: PEPPHER: efficient and productive usage of hybrid computing systems. IEEE Micro **31**(5), 28–41 (2011)
7. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **61**(6), 810–837 (2001)
8. Chrysos, G.: Intel® Xeon Phi Coprocessor-the Architecture. Intel Whitepaper (2014)
9. Dokulil, J., Bajrovic, E., Benkner, S., Pllana, S., Sandrieser, M.,Bachmayer, B.: High-level support for hybrid parallel execution of C++ applications targeting Intel Xeon Phi coprocessors. In: ICCS. Procedia Computer Science, vol. 18, pp. 2508–2511. Elsevier (2013)
10. Duran, A., Ayguadé, E., Badia, R.M., Labarta, J., Martinell, L., Martorell, X., Planas, J.: OmpSs: a proposal for programming heterogeneous multi-core architectures. Parallel Process. Lett. **21**(02), 173–193 (2011)
11. Grewe, D., O'Boyle, M.F.P.: A static task partitioning approach for heterogeneous systems using OpenCL. In: Knoop, J. (ed.) CC 2011. LNCS, vol. 6601, pp. 286–305. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19861-8_16
12. Kessler, C.W., Dastgeer, U., Thibault, S., Namyst, R., Richards, A., Dolinsky, U., Benkner, S., Traff, J.L., Pllana, S.: Programmability and performance portability aspects of heterogeneous multi-/manycore systems, pp. 1403–1408. IEEE (2012)
13. Khan, F., Han, Y., Pllana, S., Brezany, P.: Estimation of parameters sensitivity for scientific workflows. In: 2009 International Conference on Parallel Processing Workshops, pp. 457–462, September 2009
14. Khan, F., Han, Y., Pllana, S., Brezany, P.: An ant-colony-optimization based approach for determination of parameter significance of scientific workflows. In: 2010 24th IEEEInternational Conference on Advanced Information Networking and Applications (AINA), pp. 1241–1248, April 2010
15. Kołodziej, J., Khan, S.U.: Data scheduling in data grids and data centers: a short taxonomy of problems and intelligent resolution techniques. In: Nguyen, N.-T., Kołodziej, J., Burczyński, T., Biba, M. (eds.) Transactions on Computational Collective Intelligence X. LNCS, vol. 7776, pp. 103–119. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38496-7_7
16. Liu, Y., Pan, T., Aluru, S.: Parallel pairwise correlationcomputation on intel xeon phi clusters. arXiv preprint arXiv:1605.01584 (2016)
17. Luk, C.K., Hong, S., Kim, H.: Qilin: exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In: 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009, MICRO-42, pp. 45–55. IEEE (2009)

18. Memeti, S., Pllana, S.: PaREM: a novel approach for parallel regular expression matching. In: 17th International Conference on Computational Science and Engineering (CSE 2014), pp. 690–697, December 2014
19. Memeti, S., Pllana, S.: Accelerating DNA sequence analysis using Intel Xeon Phi. In: PBio at the 2015 IEEE International Symposiumon Parallel and Distributed Processing with Applications (ISPA). IEEE (2015)
20. Memeti, S., Pllana, S.: Analyzing large-scale DNA sequences on multi-core architectures. In: 18th IEEE International Conference on Computational Science and Engineering (CSE 2015). IEEE (2015)
21. Nakao, M., Lee, J., Boku, T., Sato, M.: XcalableMP implementationand performance of NAS parallel benchmarks. In: Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, p. 11. ACM (2010)
22. NCBI: National Center for Biotechnology Information U.S. NationalLibrary of Medicine (2015). http://www.ncbi.nlm.nih.gov/genbank. Accessed Dec 2015
23. Odajima, T., Boku, T., Hanawa, T., Lee, J., Sato, M.: GPU/CPU work sharing with parallel language XcalableMP-dev for parallelized accelerated computing. In: 2012 41st International Conference on Parallel Processing Workshops (ICPPW), pp. 97–106. IEEE (2012)
24. Pllana, S., Benkner, S., Xhafa, F., Barolli, L.: Hybrid performance modeling and prediction of large-scale computing systems. In: International Conference on Complex, Intelligent and Software Intensive Systems, 2008, CISIS 2008, pp. 132–138, March 2008
25. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing, 3rd edn. Cambridge University Press, Cambridge (2007)
26. Ravi, V.T., Agrawal, G.: A dynamic scheduling framework for emerging heterogeneous systems. In: 2011 18th International Conference on High Performance Computing (HiPC), pp. 1–10. IEEE (2011)
27. Scogland, T.R.W., Feng, W., Rountree, B., Supinski, B.R.: CoreTSAR: adaptive worksharing for heterogeneous systems. In: Kunkel, J.M., Ludwig, T., Meuer, H.W. (eds.) ISC 2014. LNCS, vol. 8488, pp. 172–186. Springer, Heidelberg (2014). doi:10.1007/978-3-319-07518-1_11
28. Viebke, A., Pllana, S.: The potential of the Intel (R) Xeon Phi forsupervised deep learning. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), pp. 758–765. IEEE (2015)