Predictive Resource Management for Next-generation High-Performance Computing Heterogeneous Platforms

Giuseppe Massari, Anna Pupykina, Giovanni Agosta, and William Fornaciari

Politecnico di Milano – DEIB, via Ponzio 34/5, Milano, Italy
{name.surname}@polimi.it

Abstract. High-Performance Computing (HPC) is rapidly moving towards the adoption of nodes characterized by an heterogeneous set of processing resources. This has already shown benefits in terms of both performance and energy efficiency. On the other side, heterogeneous systems are challenging from the application development and the resource management perspective. In this work, we discuss some outcomes of the MANGO project, showing the results of the execution of real applications on a emulated deeply heterogeneous systems for HPC. Moreover, we assessed the achievements of a proposed resource allocation policy, aiming at identifying a priori the best resource allocation options for a starting application.

Keywords: HPC · Heterogeneous Systems · Resource Management.

1 Introduction

Thanks to the major efforts brought on by the industrialised countries, in the upgrade of their High-Performance Computing (HPC) infrastructures, *Exascale computing* is becoming a closer reality. For example, the Summit supercomputer, for which a theoretical peak performance of 200 Petaflops has been declared, on November 2018 scored 143 Petaflops on the High Performance Linpack benchmark [1]. Although Exascale computing promises a major increase of available computational capabilities, to solve scientific and industrial challenges, the management of HPC infrastructures is growing in complexity. This is due to both the sheer size, the massive energy requirements (Summit nears the 10 MW power envelope) and the architectural complexity introduced by the presence of heterogeneous computing elements (such as CPUs, GPUs, or HW accelerators).

New access methods, such as Cloud HPC [29], are pushing for achieving a high utilisation level of such infrastructures, while hosting the execution of diverse applications, under the constraint of a limited power envelope. To meet such requirements, while managing to the growing complexity previously discussed, is a challenging task for which new resource management approaches

[1] www.top500.org

are required. Several efforts are ongoing in Europe to address these issue, developing appropriate tools to combine programming models and resource management [24,8]

Main contribution. In this paper, we explored the definition of multi-level policies to manage the computing resources (cores and memory). Since the policy identification can be quite time consuming, we propose a faster but simpler management policy at fine (temporal) grain, as well as a slower management policy [23] at a coarser grain. We show how the proposed approach allows a resource manager to identify a priori the best resource mapping solutions, given the application description and the topology of the target hardware node. We demonstrate the effectiveness of the proposed approach, by using a Low-Density Parity Check (LDPC) coding application, and an *Image Processing Filter*, executed on a deeply heterogeneous emulated HPC node.

Organization of the paper. The rest of this paper is organised as follows. In Section 2 we briefly introduce the target heterogeneous architecture and the resource management support developed and exploited for the application execution. In Section 3 we describe the proposed multi-level policy, while in Section 4 we assess its effectiveness through an experimental campaign on an emulated HPC prototype. Finally, in Section 5 we review related approaches in the state of the art, and in Section 6 we draw some conclusions and highlight future research directions.

2 Background: HPC and the MANGO Project

2.1 Deeply Heterogeneous Architectures

To achieve the necessary performance/watt figures in future Exascale HPC systems, architectural heterogeneity has been widely proposed. Its effectiveness is already demonstrated by the large number of heterogeneous systems listed in the Top500 [2] and Green500 [3] lists. In particular, the Green500, which focuses on performance/watt rather than pure performance, is dominated by heterogeneous systems, typically coupling general purpose multi-core CPUs with accelerators such as GPGPUs. Reconfigurable accelerators then, have been proposed as a further step for improving the capabilities and introduce flexibility in HPC infrastructures [15,14].

The MANGO project [10] aimed at exploring future architectures exhibiting even more heterogeneity. In MANGO, the general-purpose cores associated to a node (GN) are supported by an heterogeneous set of processing units (including multi/many-core and accelerators), forming an *Heterogeneous Node (HN).* Considering that 1) the HN can include multiple memory nodes, and 2) the processing units are interconnected through a Network-on-Chip (NoC) [9,31,32], it follows that performing resource allocation is a critical task. On architectures like this in fact, we may experience significant differences among resource allocation solutions, in terms of performance and, of course, power consumption.

[2] www.top500.org
[3] www.green500.org

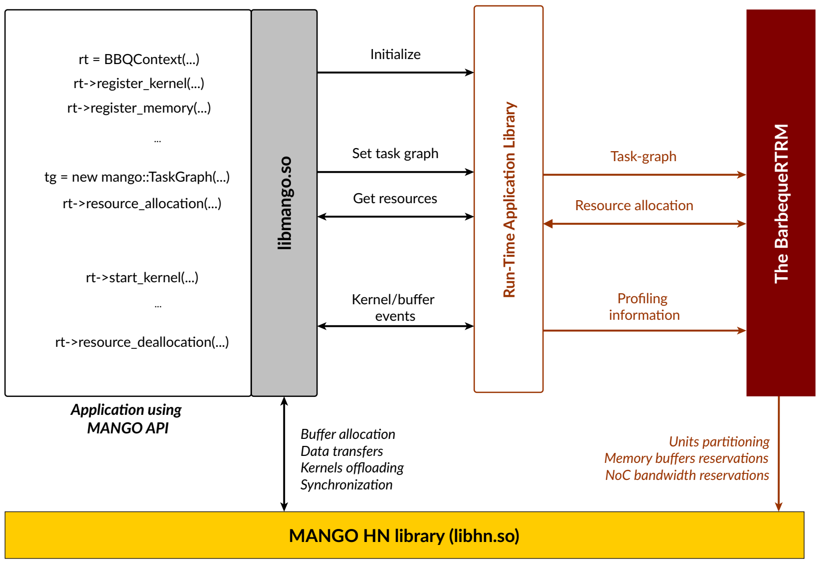


Fig. 1: MANGO Programming Model and the BarbequeRTRM integration.

2.2 Resource Management

The resource management infrastructure developed for supporting heterogeneous platforms explored in MANGO, is characterized by a tight integration between the programming model and the resource manager daemon. Through the programming model, the developer can build a task-graph based description of the application, along with per-task performance requirements. This description is then made available to the resource manager, which can lift the developer from the burden of implementing a logic for mapping the application's tasks (or kernels) and offloading them to the specific processing unit. As well as, mapping the memory buffers, needed to exchange data between tasks, onto the suitable HN-side memory nodes [2]. The resource manager is therefore aware of the system requirements and constraints, as well as of all the applications requirements. Accordingly, it can allocate resources taking into account both the system status, from the hardware perspective, and the application requirements and priorities.

In Figure 1, we sketched the integration between programming model and resource manager (the *BarbequeRTRM*) [3,25,26]. The application uses the API provided by the programming library (*libmango*) to build the task-graph based description. This is sent to the resource manager by using a specific function call. An intermediate *Run-Time Application Library* is then responsible of managing the communication flow between application and resource manager. Once the task-graph is available on the resource manager side, the mapping policy is invoked. After the policy execution, the task-graph encloses the resource mapping information, that will be exploited by the programming library to transparently perform task offloading and buffer allocation, by using the API provided by the underlying *HN library (libhn)* [7].

3 Hierarchical Policy Definition and Update

3.1 Design-time driven resource mapping

Problem Statement. We are given a HN topology, $H = \{U_f, U_b, M_f, M_b\}$, where U_f and U_b indicates the set of free and busy units, respectively, M_f and M_b indicates the set of free and busy memory units and task-graph $Tg =< B, K >$ where B is a set of requested memory buffers and K is a set of kernels. For each buffer $b \in B$ of size $S(b)$, there is a kernel or set of kernels $K(b)$ which uses b (e.g. kernel read buffer $k \overset{r}{\to} b$ and/or write to buffer $k \overset{w}{\to} b$). For each kernel $k \in K$ there is a set of preferred target processing architectures $Arch_{prefs}(k) =< Arch^0, ..., Arch^l >$ that is noted by developer. Each application has a specific priority level $appl = \{appl_h, appl_n\}$, where the high priority application $appl_h$ needed to be allocated with the requested QoS on the current HN, and the normal priority application $appl_n$ could be rescheduled on the another HN.

We aim to find all possible partitions $P = \{< M, U >_0, ..., < M, U >_m\}$ appropriate to allocate b on HN, where M is a memory unit of size $S(M)$, $M \in M_f$, $U \in U_f$ and $\forall k_i \in K\ \exists u_j \in U$ that able to execute $k_i(Arch(u_j) \in Arch_{prefs}(k_i))$, and range them by the criteria $C = \{C^m, C^{prefs}\}$. The criterion C^m defines the memory-kernel characteristics in order to select the best memory modules and includes the following specifications:

- bandwidth between the allocated processing units and the memory module;
- distance between the allocated processing units and the memory module(in hops);
- direction of data transfer (in/out);
- available space on the memory module.

More criteria could be added depending on the application requirements. The criterion C^{prefs} defines the level of the processing unit l_{arch} in $Arch_{prefs}$.

In this paper, we focused on the design-time exploration of the best units and memory nodes mapping solutions. Given the size of the solution space in fact, the time needed to find good solutions, can be often too long for considering the execution of the policy at run-time. The BarbequeRTRM allows us to perform this exploration and insert the set of mapping solutions found into a specific file, called *recipe* [20]. This file is used to specify both the per-task requirements and, optionally, a set of resource mapping solutions that the resource manager should consider at run-time.

All combinations of the preferred processing architectures can be found by following a brute-force exploration. As well as all the possible mappings of the kernel to the specific unit. However, since this approach can be time consuming, other than leading to find a redundant set of mapping solutions, we propose a heuristic policy, based on the exploitation of historical data about the previous

ALGORITHM 1: Simulation based heuristic units mapping

```
Data: Task-graph $Tg =< B, K >$, a set of preferred accelerators architectures
      $Arch_{prefs}(K) =< Arch^0, ..., Arch^l >$, a topology $H = \{U_f, U_b, M_f, M_b\}$
Result: an ordered set of partitions $P = \{< M, U >_0, ..., < M, U >_m\}$
1 $Buffers \Leftarrow BruteForce(H, B_f)$;
2 $Archs \Leftarrow BruteForce(Tg, Arch_{prefs})$ ;
3 foreach $a_i \in Archs$ do
4     $Units \Leftarrow FindAvailableUnits(U_f, a_i)$ ;
5     foreach $u_i \in Units$ do
6         if $u_i \notin P$ then
7             $b_i \Leftarrow Select(Buffers)$;
8             $P \Leftarrow newPartition(u_i, b_j)$ ;
9 $P \Leftarrow RangePartitions(P, Scores(P))$ ;
```

application executions. The heuristic goal is to limit the number of the resource partitions to consider for the allocation, on the basis of the minimal mean distance to the memory unit. The pseudo-code is reported in Algorithm 1. The first line looks over all possible buffers allocations on memory units. Line 2 creates a set of all task mappings to preferred architectures. At lines 3 and 4, for each possible combination of architectures, the sets of free units U_f of a particular HN topology are searched. Next, at lines 5-8 for each unit set that is not already included, a new partition is created with the memory mapping selected as a best of possible mappings. At the end, all partitions are sorted. Buffer mapping and partition ranging are based on the fuzzy multi-criteria analysis, with pairwise comparison of the memory-unit specifications along with the memory usage prediction in the simulation based approach. In general, simulation based approaches update a *partition score* on the basis of continuously updated information about the state of the system resources. By predicting the future state of resources we can improve the quality of the resource allocation decisions [1]. The overview of partition evaluation algorithm is presented in Algorithm 2. The Algorithm 2 first evaluates each buffer across all mappings (lines 2 and 3) by accumulating kernel-memory characteristics for each kernel what reads and/or writes to this buffer (lines 4-8). Some of these characteristics change during simulation (e.g., available bandwidth) or are constant (e.g., distance in hops). At line 9 all kernel-memory characteristics are sent to a fuzzy multi-criteria analysis [23] supplemented by the calculation of the resources utilization prediction. Lines 10-11 multiplies the scores of the current buffer with the scores of the previously evaluated buffers. In the next step, Algorithm 2 for each partition (line 12) calculates the score of the allocation according to the unit architecture in the list of preferred architectures (lines 13-15). After that, at line 16–17 the algorithm attempts to allocate and deallocate buffers (without changing statistics and calculating prediction). On allocation failure, at line 19 the score changes to indicate the memory segmentation. On the allocation success, score is normalized (line 21). Finally, at line 22 the overall score is calculated.

ALGORITHM 2: Partitions evaluation

```
Data: Task-graph $Tg =< B, K >$, a set of partitions
      $P = \{< M, U >_0, ..., < M, U >_m\}$, a set of preferred processing architectures
      $Arch_{prefs}(K) =< Arch^0, ..., Arch^l >$
Result: Scores $s = \{s_0, ..., s_m\}$
1  $s[] \Leftarrow 1.0$ ;
2  foreach $b_i \in B$ do
3      foreach $p_j \in P$ do
4          $m_i \Leftarrow p_j(b_i)$;
5          foreach $k$ such that $\exists k \xrightarrow{r} b_i$ do
6              $prop_r(p_j) \Leftarrow GetProperties(m_i, p_j(k))$ ;
7          foreach $k$ such that $\exists k \xrightarrow{w} b_i$ do
8              $prop_w(p_j) \Leftarrow GetProperties(m_i, p_j(k))$;
9      $eval(b_i)[] \Leftarrow BufferAnalysis(m_i, prop_r, prop_w)$;
10     foreach $e \in eval(b_i)$ do
11         $s[i] \Leftarrow s[i] \times e$ ;
           /* with keeping negative scores to indicate the predicted usage of
              $m_i$ by $appl_h$                                              */
12 foreach $p_i \in P$ do
13     $score_{prefs} \Leftarrow 1$ ;
14     foreach $u \in p_i$ do
15         $score_{prefs} \Leftarrow score_{prefs} \times n_k^{level(Arch_{prefs})}$ ;
16     $err \Leftarrow Allocate(p_i)$;
17     $Deallocate(p_i)$;
18     if $err! = Success$ then
19         $s[i] \Leftarrow OutOfMemory$ ;
20     else
21         $s[i] \Leftarrow Normalise(s[i])$;
22     $s[i] \Leftarrow s[i] \times 100 \div score_{prefs}$;
```

4 Experimental Evaluation

4.1 Hardware Setup

The experimental hardware platform consists of a FPGA-based prototype, on which we deployed an heterogeneous set of custom processors, distributed in tiles interconnected through a 2D-mesh Network-on-Chip (NoC). The system also includes multiple memory nodes, each attached to a different tile. In the MANGO project, we explored several possible hardware configurations. For our experimental evaluations, we considered the one shown in Figure 2. This includes 8 processing units: 3 dual-core PEAK processors (MIPS), 2 GPU-like units for SIMD executions (NU+) and 3 HW accelerators for vectors and image processing. This configuration includes two memory nodes, attached to tiles 0 and 1.

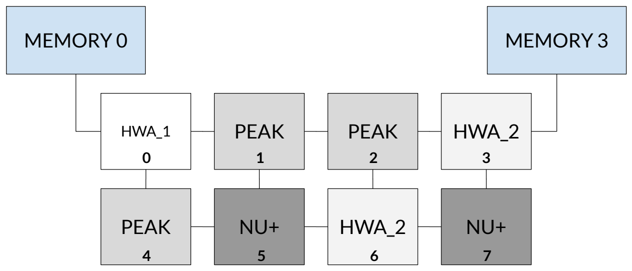


Fig. 2: Topology of the MANGO platform (single Heterogeneous Node) used for the experimental evaluations.

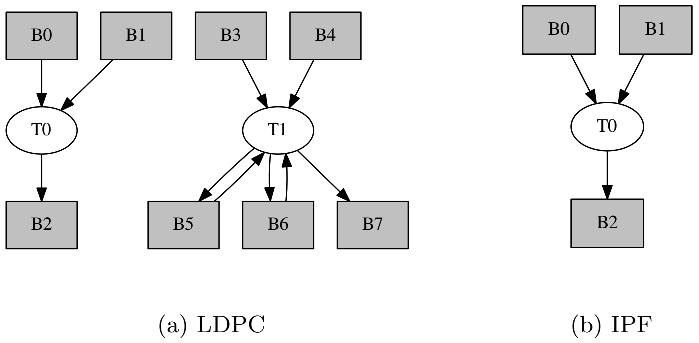
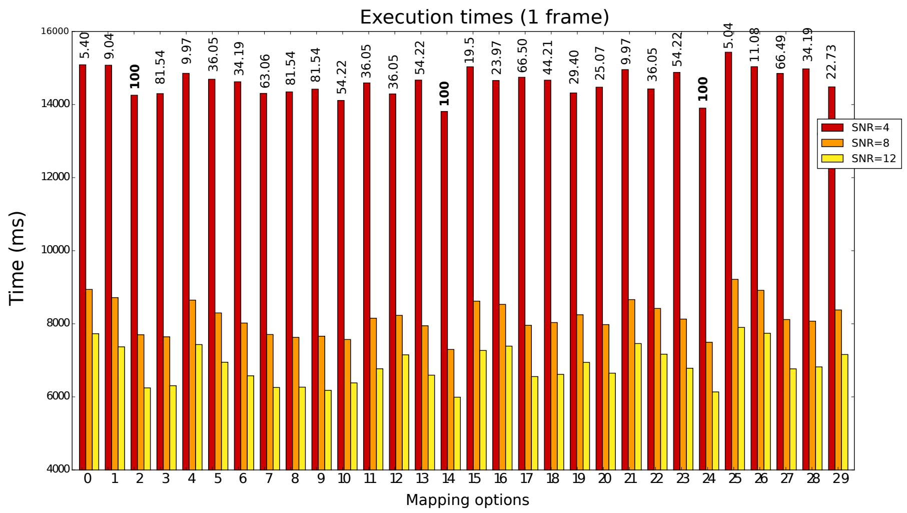


Fig. 3: Task-graph based description of the test applications.

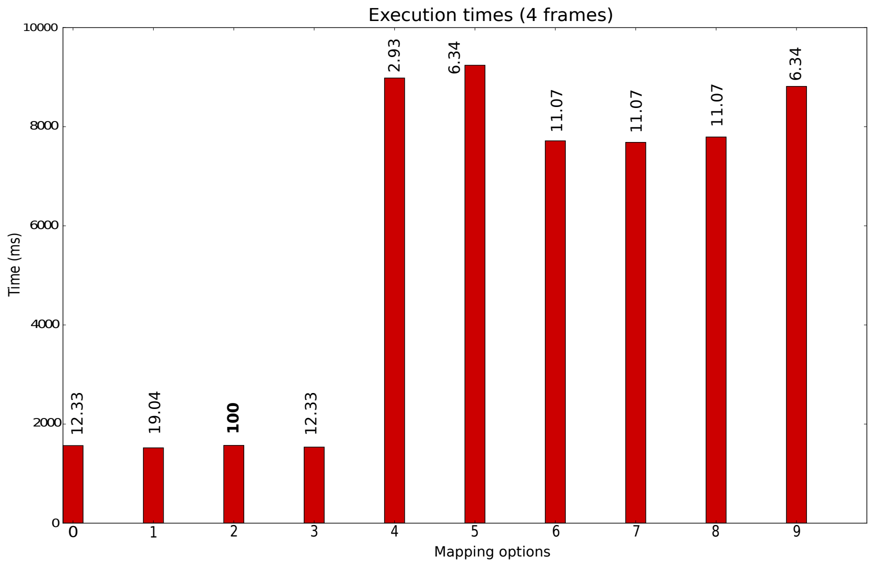
4.2 Test Applications

In this work, we used a *Low-Density Parity Check (LDPC)* application and an *Image Processing Filter*. In particular, LDPC is a type of linear error correction algorithm, devised in the 1960s [12], that have become practical to implement only since the 1990s [19], due to its high computational requirements.

In Figure 3 we observe the task-graph based description of the two applications. For LDPC, the resource allocation policy must find good mapping solutions for the 8 buffers and the offload of 2 kernels. The kernel binaries are available only for architecture PEAK. The Image Processing Filter instead, uses 2 input buffer to store frames coming from two streams, plus 1 output buffer to store the result. The processing is performed by a single kernel, for which two mapping options are available: PEAK and HW accelerators (HWA_2).



(a) LDPC



(b) IFS

Fig. 4: Execution times for each explored mapping solution. The LDPC SNR input parameter has been set to 4, 8 and 12 (no noise). The Image Processing Filter has been executed for processing streams of 4 frames.

4.3 Experimental Results

The goal of our experiments has been two-fold: *1)* observe how the platform topology actually affects the performance of the applications and *2)* verify the effectiveness of the proposed heuristic in predicting good resource mapping solutions, a priori. We explored the set of resource mapping solutions, shown in Table 1, for the two applications, by executing them target platform and enforcing the mapping through the resource manager. For example, the resource mapping solution "0" for LDPC consists of mapping the two kernels on PEAK processors located respectively at tiles 1 and 4, and all the buffers on memory 3.

In Figure 4a, we reported all the average execution times of the LDPC application, needed to process a single input frame, for three different of the SNR parameter ($SNR = \{4, 8, 12\}$). Although the kernels have been executed on the same type of processors, we experienced different (average) execution times, which lead us to conclude that, as expected, the platform topology actually impacts on while the input parameter assumes different values. Regarding the predictions made by the proposed policy, on top of the bars we reported the score computed by the heuristic, for each of the tested mapping options. The higher the score the better the mapping solution. We can observe that the predictive model actually found the solutions 2, 14 and 24 as the best options. To the contrary, solutions 0, 1, 4 and 25 were expected to be the worst performing. In the real executions, we verify whether the prediction was valid.

In Figure 4b, we can observe the same kind of tests performed with the Image Processing Filter. The resource mapping options from 0 to 3 refer to the execution of the kernel on the HW accelerator, while from 4 to 9, we have the mapping options for which the kernel is offloaded on a PEAK processor. Trivially, the best resource mapping option is found among the ones including the HW accelerator (option 2). The rationale behind this result is that the policy does not have enough information about the speed-up introduced by HW accelerators, with respect to the programmable accelerator. As a result, the scores of solutions 0, 1 and 3 are lower than they should be. More interesting is to observe how for the options mapping the kernel onto a PEAK, the set 6, 7 and 8 were expected to be the best ones. Looking at Table 1, these are the options according to which we map the buffers into the memory node closest to the processor. The policy is therefore quite effective in capturing the characterization of the system topology.

Overall, the proposed heuristic succeeded in predicting the boundaries of the solutions space, i.e, best and worst resource mappings. For intermediate solutions, the score does not always match the real performance, but in most of the cases, the prices paid for the misprediction is negligible.

5 Related Works

Traditionally, resource management in HPC is limited by assigning to each application a set of physical nodes at the job scheduler level taking into account different aspects of the cluster architecture, such as the topology of the machine

Table 1: Explored resource mapping solutions for LDPC application running on the target MANGO platform.

LDPC			IFS		
Solution ID	Processors	Memories	Solution ID	Processors	Memories
0	1 4	3 3 3 3 3 3 3 3	0	3	0 0 0
1	4 1	3 3 3 3 3 3 3 3	1	6	3 3 3
2	1 4	0 0 0 0 0 0 0 0	2	3	3 3 3
3	4 1	0 0 0 0 0 0 0 0	3	6	0 0 0
4	1 4	0 0 0 3 3 3 3 3	4	4	3 3 3
5	4 1	0 0 0 3 3 3 3 3	5	1	3 3 3
6	1 4	3 3 3 3 0 0 0 0	6	2	3 3 3
7	4 1	0 0 3 0 0 0 0 0	7	4	0 0 0
8	4 1	0 0 0 0 3 0 0 0	8	1	0 0 0
9	1 4	0 0 3 0 0 0 0 0	9	2	0 0 0
10	1 2	3 3 3 3 3 3 3 3			
11	2 1	3 3 3 3 3 3 3 3			
12	1 2	0 0 0 0 0 0 0 0			
13	2 1	0 0 0 0 0 0 0 0			
14	1 2	0 0 0 3 3 3 3 3			
15	2 1	0 0 0 3 3 3 3 3			
16	1 2	3 3 3 3 0 0 0 0			
17	2 1	0 0 3 0 0 0 0 0			
18	2 1	0 0 0 0 3 0 0 0			
19	1 2	0 0 3 0 0 0 0 0			
20	4 2	3 3 3 3 3 3 3 3			
21	2 4	3 3 3 3 3 3 3 3			
22	4 2	0 0 0 0 0 0 0 0			
23	2 4	0 0 0 0 0 0 0 0			
24	4 2	0 0 0 3 3 3 3 3			
25	2 4	0 0 0 3 3 3 3 3			
26	4 2	3 3 3 3 0 0 0 0			
27	2 4	0 0 3 0 0 0 0 0			
28	2 4	0 0 0 0 3 0 0 0			
29	4 2	0 0 3 0 0 0 0 0			

to determine the best choice among the available nodes based upon their position within the network [13], or emphasizing various targets, such as power-awareness [22] or resilience-awareness [5]. More recently, resource management has focused on the specific type of applications, such as MapReduce-based applications. A widely used cluster resource managers in the Hadoop system, e.g. YARN [27] or Mesos [16], allow allocating resources, such as CPU and memory, to multiple big data applications. However, none of them can directly support the management of the deeply heterogeneous resources, out-of-box. Based on the YARN framework, the resource management strategy and scheduling mechanism to suit for the heterogeneous CPU-FPGA cluster was proposed in [17]. This approach modifies the resource representation scheme that manages logical FPGA accelerator functionality for better scheduling and provides development interfaces for easily usage of FPGAs. An heterogeneous ARM/FPGA SoC was considered as the target architecture for the power capping technique proposed in [28]. This approach combines power capping with coordinated dynamic voltage and frequency scaling (DVFS), data partitioning and core allocations for efficient use of both ARM processor and streaming accelerators on FPGA concurrently. A similar approach, leveraging a hardware implementation for the power capping, was proposed in [30,33], based on the modeling of DVFS and power gating actuators provided in [34]. A run-time task allocator for heterogeneous many-core platforms, SPARTA, was presented in [6]. It uses the variability in workload memory and computational requirements in order to provide energy efficient task-to-core allocations. SPARTA is proposed for a generic Linux environment and single-ISA shared memory heterogeneous multi-processing. Three resource allocation algorithms suitable for heterogeneous HPC systems focused on the efficient management of critical, accelerator-like resources were presented in [21]. In the context of heterogeneous high-end embedded systems, co-scheduling of multiple application has also been studied. A recent survey of such approaches, and a technique that leverages Linux Control Groups can be found in [18].

6 Conclusions

In this work, we briefly introduced the MANGO Project, in which we explored the architectural possibilities of next-generation HPC systems, based on a deeply heterogeneous set of processing units and multiple memory nodes. We developed a programming model integrated with a resource management framework. We proposed a heuristic-based policy to predict the best resource mapping solutions for each application, such that the resource manager can quickly pick them at run-time, without introducing additional overhead. We validated the policy by executing two real applications on a emulated heterogeneous platform prototype, enforcing different resource mapping options. We observed how the policy succeeded in the identification of the best and the worst options.

Future works will go in the direction of collecting more information about the behaviour of the applications and response of the hardware resources, especially in scenarios of resource contention. With this knowledge, we will be able to improve the policy and therefore the capabilities of the resource manager in taking decisions at run-time. Furthermore, we plan to extend the capabilities of the programming model to include dynamic recompilation of the kernels, through a partial dynamic compilation library supporting arbitrary C++ code [4].

7 Acknowledgments

This research was partially funded by the H2020 EU projects “MANGO” (grant no. 671668) and “RECIPE” (grant no. 801137 [11]).

References

1. Ababei, C., Ghorbani Moghaddam, M.: A survey of prediction and classification techniques in multicore processor systems. IEEE Transactions on Parallel and Distributed Systems pp. 1–1 (10 2018). https://doi.org/10.1109/TPDS.2018.2878699
2. Agosta, G., Fornaciari, W., Massari, G., Pupykina, A., Reghenzani, F., Zanella, M.: Managing Heterogeneous Resources in HPC Systems. In: Proc. of PARMA-DITAM '18. pp. 7–12. ACM (2018). https://doi.org/10.1145/3183767.3183769
3. Bellasi, P., Massari, G., Fornaciari, W.: Effective runtime resource management using linux control groups with the barbequertrm framework. ACM Trans. Embed. Comput. Syst. **14**(2), 39:1–39:17 (Mar 2015). https://doi.org/10.1145/2658990
4. Cherubin, S., Agosta, G.: libversioningcompiler: An easy-to-use library for dynamic generation and invocation of multiple code versions. SoftwareX **7**, 95 – 100 (2018). https://doi.org/https://doi.org/10.1016/j.softx.2018.03.006
5. Dauwe, D., Pasricha, S., Maciejewski, A.A., Siegel, H.J.: Resilience-aware resource management for exascale computing systems. IEEE Transactions on Sustainable Computing **3**(4), 332–345 (Oct 2018). https://doi.org/10.1109/TSUSC.2018.2797890
6. Donyanavard, B., Mück, T., Sarma, S., Dutt, N.: Sparta: Runtime task allocation for energy efficient heterogeneous manycores. In: 2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS). pp. 1–10 (Oct 2016)
7. Flich, J., Agosta, G., Ampletzer, P., Alonso, D.A., Brandolese, C., Cilardo, A., Fornaciari, W., Hoornenborg, Y., Kovac, M., Maitre, B., Massari, G., Mlinaric, H., Papastefanakis, E., Roudet, F., Tornero, R., Zoni, D.: Enabling HPC for QoS-sensitive applications: The MANGO approach. In: 2016 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 702–707 (March 2016)
8. Flich, J., Agosta, G., et al.: Mango: Exploring manycore architectures for next-generation hpc systems. In: 2017 Euromicro Conference on Digital System Design (DSD). pp. 478–485 (Aug 2017). https://doi.org/10.1109/DSD.2017.51
9. Flich, J., Alessandro, C., Kovač, M., Tornero, R., Martínez, J.M., Picornell, T.: Deeply heterogeneous many-accelerator infrastructure for hpc architecture exploration. In: Parallel Computing Conference (ParCo) (2017)
10. Flich, J., Agosta, G., Ampletzer, P., Alonso, D.A., Brandolese, C., Cappe, E., Cilardo, A., Dragic, L., Dray, A., Duspara, A., Fornaciari, W., Fusella, E., Gagliardi, M., Guillaume, G., Hofman, D., Hoornenborg, Y., Iranfar, A., Kovac, M., Libutti, S., Maitre, B., MartÃnez, J.M., Massari, G., Meinds, K., Mlinaric, H., Papastefanakis, E., Picornell, T., Piljic, I., Pupykina, A., Reghenzani, F., Staub, I., Tornero, R., Zanella, M., Zapater, M., Zoni, D.: Exploring manycore architectures for next-generation HPC systems through the MANGO approach. Microprocessors and Microsystems **61**, 154 – 170 (2018). https://doi.org/https://doi.org/10.1016/j.micpro.2018.05.011
11. Fornaciari, W., Agosta, G., Atienza, D., Brandolese, C., Cammoun, L., Cremona, L., Cilardo, A., Farres, A., Flich, J., Hernandez, C., Kulchewski, M., Libutti, S., Martínez, J.M., Massari, G., Oleksiak, A., Pupykina, A., Reghenzani, F., Tornero, R., Zanella, M., Zapater, M., Zoni, D.: Reliable power and time-constraints-aware predictive management of heterogeneous exascale systems. In: Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. pp. 187–194. SAMOS '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3229631.3239368

12. Gallager, R.: Low-density parity-check codes. IRE Transactions on information theory **8**(1), 21–28 (1962)
13. Georgiou, Y., Jeannot, E., Mercier, G., Villiermet, A.: Topology-aware resource management for hpc applications. In: Proceedings of the 18th International Conference on Distributed Computing and Networking. pp. 17:1–17:10. ICDCN '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3007748.3007768
14. Georgopoulos, K., Mavroidis, I., Lavagno, L., Papaefstathiou, I., Bakanov, K.: Energy-efficient heterogeneous computing at exascale – ecoscale. In: Hardware Accelerators in Data Centers, pp. 199–213. Springer (2019)
15. Herbordt, M.C., VanCourt, T., Gu, Y., Sukhwani, B., Conti, A., Model, J., DiSabello, D.: Achieving high performance with fpga-based computing. Computer **40**(3) (2007)
16. Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R., Shenker, S., Stoica, I.: Mesos: A platform for fine-grained resource sharing in the data center. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. pp. 295–308. NSDI'11, USENIX Association, Berkeley, CA, USA (2011), http://dl.acm.org/citation.cfm?id=1972457.1972488
17. Li, R., Yang, Q., Li, Y., Gu, X., Xiao, W., Li, K.: Heteroyarn: A heterogeneous fpga-accelerated architecture based on yarn. IEEE Transactions on Parallel and Distributed Systems pp. 1–1 (2019). https://doi.org/10.1109/TPDS.2019.2905201
18. Libutti, S., Massari, G., Fornaciari, W.: Co-scheduling tasks on multi-core heterogeneous systems: An energy-aware perspective. IET Computers Digital Techniques **10**(2), 77–84 (2016). https://doi.org/10.1049/iet-cdt.2015.0053
19. MacKay, D.J., Neal, R.M.: Near shannon limit performance of low density parity check codes. Electronics letters **32**(18), 1645 (1996)
20. Massari, G., Paone, E., Bellasi, P., Palermo, G., Zaccaria, V., Fornaciari, W., Silvano, C.: Combining application adaptivity and system-wide resource management on multi-core platforms. In: 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV). pp. 26–33 (July 2014). https://doi.org/10.1109/SAMOS.2014.6893191
21. Netti, A., Galleguillos, C., Kiziltan, Z., Sîrbu, A., Babaoglu, O.: Heterogeneity-aware resource allocation in hpc systems. In: Yokota, R., Weiland, M., Keyes, D., Trinitis, C. (eds.) High Performance Computing. pp. 3–21. Springer International Publishing, Cham (2018)
22. Patki, T., Lowenthal, D.K., Sasidharan, A., Maiterth, M., Rountree, B.L., Schulz, M., de Supinski, B.R.: Practical resource management in power-constrained, high performance computing. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing. pp. 121–132. HPDC '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2749246.2749262
23. Pupykina, A., Agosta, G.: Optimizing memory management in deeply heterogeneous hpc accelerators. In: 2017 46th International Conference on Parallel Processing Workshops (ICPPW). pp. 291–300 (Aug 2017). https://doi.org/10.1109/ICPPW.2017.49
24. Silvano, C., Agosta, G., et al.: The antarex tool flow for monitoring and autotuning energy efficient hpc systems. In: 2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS). pp. 308–316 (July 2017). https://doi.org/10.1109/SAMOS.2017.8344645
25. Silvano, C., Fornaciari, W., Crespi Reghizzi, S., Agosta, G., et al.: 2parma: Parallel paradigms and run-time management techniques for many-core architectures. In: 2010 IEEE Computer Society Annual Symposium on VLSI. pp. 494–499 (July 2010). https://doi.org/10.1109/ISVLSI.2010.93

26. Silvano, C., Fornaciari, W., Crespi Reghizzi, S., Agosta, G., et al.: Parallel paradigms and run-time management techniques for many-core architectures: The 2parma approach. In: 2011 9th IEEE International Conference on Industrial Informatics. pp. 835–840 (July 2011). https://doi.org/10.1109/INDIN.2011.6035001
27. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., Baldeschwieler, E.: Apache hadoop yarn: Yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing. pp. 5:1–5:16. SOCC '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2523616.2523633
28. Wu, Y., Nikolopoulos, D.S., Woods, R.: Runtime support for adaptive power capping on heterogeneous socs. In: 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS). pp. 71–78 (July 2016). https://doi.org/10.1109/SAMOS.2016.7818333
29. Ziegler, W., D'ippolito, R., D'Auria, M., Berends, J., Nelissen, M., Diaz, R.: Implementing a "one-stop-shop" providing smes with integrated hpc simulation resources using fortissimo resources. In: eChallenges e-2014, 2014 Conference. pp. 1–11. IEEE (2014)
30. Zoni, D., Cremona, L., Fornaciari, W.: All-digital energy-constrained controller for general-purpose accelerators and cpus. IEEE Embedded Systems Letters pp. 1–1 (2019). https://doi.org/10.1109/LES.2019.2914136
31. Zoni, D., Flich, J., Fornaciari, W.: Cutbuf: Buffer management and router design for traffic mixing in vnet-based nocs. IEEE Transactions on Parallel and Distributed Systems **27**(6), 1603–1616 (June 2016). https://doi.org/10.1109/TPDS.2015.2468716
32. Zoni, D., Canidio, A., Fornaciari, W., Englezakis, P., Nicopoulos, C., Sazeides, Y.: Blackout: Enabling fine-grained power gating of buffers in network-on-chip routers. J. Parallel Distrib. Comput. **104**, 130–145 (2017). https://doi.org/10.1016/j.jpdc.2017.01.016
33. Zoni, D., Cremona, L., Cilardo, A., Gagliardi, M., Fornaciari, W.: Powertap: All-digital power meter modeling for run-time power monitoring. Microprocessors and Microsystems - Embedded Hardware Design **63**, 128–139 (2018). https://doi.org/10.1016/j.micpro.2018.07.007
34. Zoni, D., Fornaciari, W.: Modeling dvfs and power-gating actuators for cycle-accurate noc-based simulators. J. Emerg. Technol. Comput. Syst. **12**(3), 27:1–27:24 (Sep 2015). https://doi.org/10.1145/2751561