Towards Energy Efficient Orchestration of Cloud Computing Infrastructure

A. Scionti, K. Goga, F. Lubrano, O. Terzo

Abstract The emerging of new Cloud services and applications demanding for ever more performance (i.e., on one hand, the rapid growth of applications using deep learning –DL, on the other hand, HPC-oriented work-flows executed in Cloud) is continuously putting pressure on Cloud providers to increase capabilities of their large data centers, by embracing more advanced and heterogeneous devices [2, 3, 11]. Hardware heterogeneity also helps Cloud providers to improve energy efficiency of their infrastructures by using architectures dedicated to specific workloads. However, heterogeneity represents a challenge from the infrastructure management perspective. In this highly dynamic context, workload orchestration requires advanced algorithms to not defeat the efficiency provided by the hardware layer. Despite past works partially addressed the problem, a comprehensive solution is still missing.

This paper presents the solution studied within the European H2020 project OPERA [1]. Our approach is intended for managing the workload in large infrastructures running heterogeneous systems, by using a two-steps approach. Whenever new jobs are submitted, an energy-aware allocation policy is used to select the most efficient nodes where to execute the incoming jobs. In a second step, the whole workload is consolidated by means of the optimization of a cost model. This paper focuses on an allocation algorithm aimed at reducing the overall energy consumption; it also presents the results of simulations on a State-of-the-Art framework. When compared with well-known and broadly adopted allocation strategies, the proposed approach results in a tangible energy-saving (up to 30% compared to First Fit allocation policy, and up to 45.2% compared to the Best Fit), thus demonstrating energy efficiency superiority.

A. Scionti, K. Goga, F. Lubrano, O. Terzo
Istituto Superiore Mario Boella (ISMB), e-mail: {scionti,goga,lubrano,terzo}@ismb.it

1 Introduction

The growing importance for converged platforms, i.e., computing infrastructures which software and hardware stack is optimized for running both compute-centric (also referred to as HPC-oriented) and data-centric applications, is gaining momentum. Cloud computing paradigm brought the opportunity to exploit massive computational resources to the masses, avoiding the costs of expensive dedicated machines. Recently, Cloud providers and hyper-scaler introduced dedicated offering based on the addition of GP-GPUs and FPGAs to better support HPC-oriented workloads. Beside the heterogeneity across CPUs families, the biggest change is represented by the introduction of GP-GPUs and FPGAs on dedicated instances. For instance, Amazon AWS- EC2 [2], Microsoft Azure [3] offer GPU-accelerated instances covering the most recently introduced architectures (i.e., Nvidia Pascal [4], Nvidia Volta [5]).

Although less common, many-cores, such as the Intel XeonPhi chip family [6], Kalray MPPA [7] and Tilera architectures [8] emerged as good contenders for scientific workloads. Their high-parallel micro-architecture can benefit from vectorization acceleration features, as well as the sharing of a common ISA with host CPUs. Besides, the recent interest in machine learning (ML) and, more specifically, in deep learning techniques (DL), raised the attention of both Cloud and hardware providers on more specialized architectures. In this context, services powered by deep neural networks (DNNs) and their variants led hardware providers to envision dedicated chips, able to largely increase the performance/Watt ratio over conventional architectures. Recently, FPGAs emerged as a candidate, thanks to the optimal trade-off between performance, overall power consumption, and flexibility.

Despite the many benefits of heterogeneity adoption, its integration in the Cloud software stack poses several challenges [9]. Workload orchestration is a popular Cloud approach to provide management features over: *i*) reserving computing resources for the incoming jobs (i.e., virtual machines –VMs– to launch); *ii*) dynamically adapting the workload distribution to satisfy some specified criteria. In fact, the abundance of different hardware devices, each exposing peculiar features, requires a careful placement of the incoming VMs through more intelligent and flexible tools for orchestrating hybrid workloads.

Power(energy) management within allocation algorithms was firstly introduced to minimize the number of running servers [21]. A server in idle state consumes up to 65% of its peak power, thus, power consumption can be minimized by concentrating the workload on the minimum number of physical nodes and switching off unused ones. However, this solution requires to handle application throughput and execution time constraints, as defined in SLA. Following this approach, authors in [17] have proposed heuristics for dynamic adaptation of a VMs' allocation policy at run-time. According to the current resource utilization, their heuristic applies live migration and idle nodes sleep mode switching. The aforementioned approach was tested on a data center composed of "heterogeneous" machines and took into account the characteristics of each physical node to reduce the energy consumption. Despite several works tackled the problem of integrating power(energy) awareness

into allocation algorithms, the most used ones are still based on simple, fast approaches such as First Fit (FF) and Best Fit (BF); although, in this cases, heterogeneity is not addressed. More accurate simulations using heterogeneous infrastructures are made in [22]. Here, heterogeneity concerns hardware resources such as GP-GPUs, MICs and FPGAs in addition to traditional CPUs.

This paper presents a comprehensive study of an energy-aware orchestration approach, as formulated in the OPERA project [1] to address the challenges related to the dynamic nature of work-flows. The ECRAE –Efficient Cloud Resources Allocation Engine– greedy strategy is proposed to fast and effectively allocate incoming VMs to the most suitable node. Furthermore, ECRAE takes into account heterogeneity provided by acceleration devices. When compared to traditional and widely adopted allocation policies (e.g., First Fit and Best Fit), the ECRAE solution is better by far, allowing to save up to $\sim$45% of energy. To summarize, the main contribution are: *i*) providing a detailed insight of the two-steps approach devised by OPERA project for the orchestration of workloads in a heterogeneous infrastructure; *ii*) describing the performance-power consumption model for acceleration devices used to drive the orchestrator; and *iii*) comparing the performance of the ECRAE allocation policy on different workload conditions, with traditional allocation policies.

The remainder of this work is organized as follows. Section 2 provides a description of the most relevant technologies in modern heterogeneous infrastructures. A detailed description of the ECRAE greedy strategy is provided in Section 3, along with the discussion on the performance and power consumption models for accelerators. Next, in section 4, after describing the simulation environment and the used workloads, we present the experimental evaluation results of the proposed strategy. Finally, section 5 concludes the work.

2 Orchestrating heterogeneous infrastructures

Aiming at supporting a new class of applications in modern data centers (i.e., deep learning and scientific computing applications), advanced architectural solutions comprising both hardware and software components at different levels must be leveraged. At the hardware level, this reflects in the massive adoption of specialized devices, as well as a large differentiation of the processor families. On the other hand, at the software level, heterogeneity reflects in emerging frameworks (e.g., TensorFlow, Caffe, etc.) and management components (e.g., Open Stack Cyborg project.). Despite these efforts, power(energy) efficiency still remain a challenge. Techniques, both hardware (e.g., dynamic voltage and frequency scaling –DVFS) and software (e.g., dynamic migration of VMs) can be applied to reduce the power consumption without negatively impacting on the performance. Resource orchestrator (RO) is a key component in Cloud data centers to efficiently manage the underlying resources, by means of always more sophisticated algorithms. RO can be defined as the set of operations that Cloud providers undertake (either manually or automatically via dedicated computer programs) for selecting, deploying, monitoring, and dynamically controlling the configuration of hardware and software resources, reflecting in a set of QoS-assured components seamlessly delivered to end users.

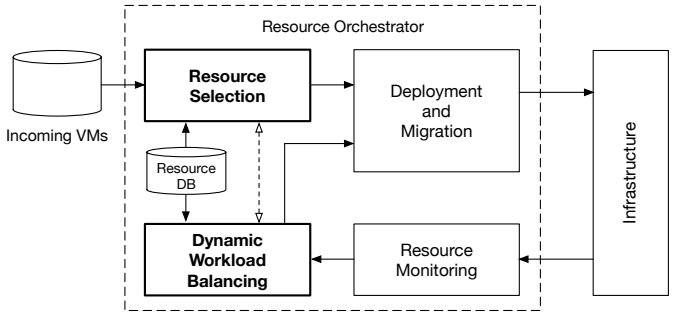

Fig. 1: Resource Orchestrator (RO): the main operation performed for dynamically controlling the Cloud infrastructure resources.

Figure 1 shows the main flow of operations performed by the RO. There are two main components that have a key role in managing the infrastructure: *i*) the resource selection (RS) module; and the *ii*) dynamic workload balancing (DWB). The former is responsible to find a match between available resources and the incoming VMs. To this end, the RS accesses to an internal database containing the updated description of the nodes in the infrastructure in terms of features (i.e., total number of cores, total amount of memory, storage and bandwidth) and the resources still available for allocating other VMs. The RS takes decisions based on a fast algorithm (e.g., ECRAE has a complexity equals to $O(n)$), taking into account allocation constraints for specific VMs. For instance, a VM requiring the use of an accelerator can be allocated only on nodes exposing that type of accelerator. Whenever the resource has been selected, a dedicated agent (deployment and migration –DM) performs all the necessary actions to boot-up the VM on the node. The dynamic nature of workloads requires a constant monitoring of the used resources. The resource monitor (RM) is in charge of acquiring all the metrics needed to measure the resource usage, and updating the internal RO database. Information gathered by RM are used by the DWB component for periodically re-balancing the whole workload. The DWB applies complex algorithms to accomplish such task, which has been demonstrated to be a NP-complete optimization problem [23]. The result of DWB algorithms is the schedule of VMs that must be migrated. The periodicity of workload re-balance depends on several factors, including the number of nodes and VMs to take into account, the complexity of the DWB algorithms, and Cloud provider specific constraints.

2.1 Infrastructure heterogeneity

A modern data center relies on several hardware and software components to ensure adequate capabilities for end user applications.

Figure 2 depicts the general architecture of a heterogeneous data center. The hardware level is composed by commodity hardware (i.e., servers, storage nodes and network switches), with server nodes augmented by accelerators. Such devices,

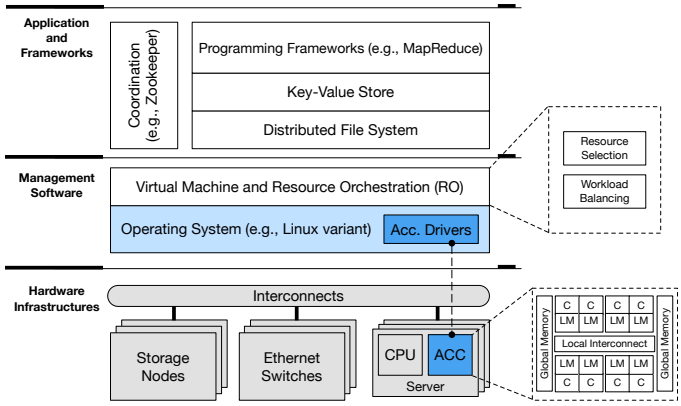

Fig. 2: Heterogeneous data center: hardware architecture comprises different types of CPUs and accelerators, while software stacks provides necessary management and programming frameworks.

typically provide large parallelism through high number of dedicated cores (C) coupled with ultra-fast local memories (LM). Data movement is ensured internally by local high-speed, low latency interconnects. Other, specialized cores can be also integrated, to ensure the highest performance.

Interfacing with the remainder of the system is enabled by global memory interface, and the driver integrated into the operating system. Global memory is implemented as a dedicated, on-board, block of fast memory. On top of the operating system, the virtualization layer is implemented, providing the functionalities that permit both to create and migrate virtual machines, and to orchestrate resources. Generally, resource orchestration is obtained by implementing the management logic in a distributed fashion. Virtualization provides the substrate for integrating in the platform a distributed file system and to enable end users to create their applications through dedicated frameworks.

Among the others, FPGAs recently gained the attention as general purpose class of accelerators. The flexibility of such software programmable chips allows them to be reused over time for totally different application purposes (e.g., on-the-fly encryption, data compression, etc.). In fact, in contrast to other accelerators, FPGAs do not provide a fixed architecture. Instead, their internal resources (i.e., local blocks of distributed memory, logic elements, flip-flops, etc.) can be arranged to resemble any digital circuit. In that, FPGA fabrics exhibit a data-flow architecture, where the data processing is quickly performed without executing any explicit instruction.

Figure 3 depicts the internal architecture of a FPGA, along with the power models experimentally derived for two Intel chips [9, 10]. In contrast to other accelerators, the data-flow oriented architecture provides an almost constant throughput on processing data, as well as the power consumption is depending on the amount of resources allocated (active) and the clock frequency on the chip. During application compilation, the FPGA tool chain generates a circuit which is mapped on fabric resources. Thus, the percentage of resources consumed, proportionally determines the overall power consumption.

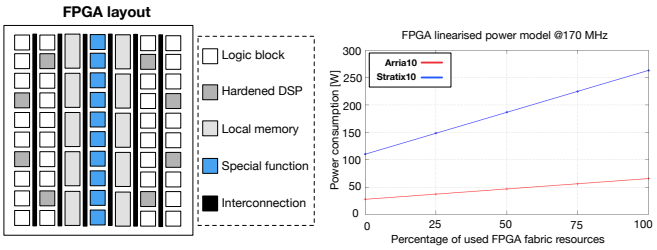

Fig. 3: On the left, the FPGA fabric resource layout. On the right, the linearized power model for the Intel Arria10 and Intel Stratix10 devices.

3 Energy-aware Cloud Resources Allocation Engine – ECRAE

The Energy-aware Cloud Resources Allocation Engine (ECRAE) is a greedy algorithm that aims at finding optimal placing of incoming VMs in heterogeneous environments. To this end, the algorithm ranks all the available nodes on their relative efficiency (i.e., power consumption weighted by the current load), and selects the most effective one that can accommodate the requested VM. In addition, ECRAE takes into account specific constraints for running the VM; for instance, some VMs in the workload may require the access to a specific accelerator (e.g., FPGA). In that case, ECRAE will possibly assign a node that exposes such device, reverting on nodes without acceleration only in case no one can accommodate the request.

Algorithm 1 shows the main steps used to select the most efficient node. Focusing on supporting FPGA acceleration (although, other kind of accelerator can also be used), first, the procedure evaluates if the VM requires acceleration via FPGA (line 2). In that case, the list of hosting nodes equipped with the required device model is used to calculate the rank (line 3). To this end, the cost of using a given node is calculated (lines 19–27). To reduce power consumption, hosts already active are firstly considered as candidates for allocation (lines 4–5); however, if all the nodes are sleeping, the procedure will select the less costly one (lines 6–7). Whenever the VM does not require to access the FPGA, the procedure looks at the list of nodes without acceleration support. Similarly to the previous case, the algorithm tries to select a node from the list of active ones (lines 10–12); otherwise, the less costly node will be selected (line 14). In all the cases, the same *CalculateRanking* cost function is used (line 19–27).

3.1 Data center power-cost model

To enable the ECRAE algorithm to take correct decisions when a VM demands for the use of an accelerator, we modeled the performance and power consumption of a generic acceleration device. To this end, we must consider that VMs come with an associated cost that is expressed by the amount of resources to access (i.e., number

Algorithm 1 ECRAE energy-aware greedy allocation algorithm.

```
1:  procedure ECRAEALLOCATIONALGORITHM(hostList, vm)
2:      if vm.supportFPGAAcceleration() then
3:          FPGAHostList ← hostList.getHostSuitableForVm(vm).getHostWithFPGA()
4:          if !FPGAHostList.getActive().isEmpty() then
5:              return calculateRanking(FPGAHostList.getActiveHost(), vm)
6:          else
7:              return calculateRanking(FPGAHostList, vm)
8:          end if
9:      else
10:         NoFPGAHostList ← hostList.getHostSuitableForVm(vm).getHostWithoutFPGA()
11:         if !NoFPGAHostList.getActive().isEmpty() then
12:             return calculateRanking(NoFPGAHostList.getActiveHost(), vm)
13:         else
14:             return calculateRanking(NoFPGAHostList, vm)
15:         end if
16:     end if
17: end procedure
18:
19: function CALCULATERANKING(hostList, vm)
20:     for host in hostList do
21:         actualPowerConsumption ← host.getPower()
22:         CPULoadIncrement ← vm.getPe/host.getPe
23:         score ← CPULoadIncrement ∗ actualPowerConsumption
24:         ranking ← (score, host)
25:     end for
26:     return ranking.minScore().getHost()
27: end function
```

of cores, amount of memory, storage, etc.), but also by the amount of instructions to be consumed. Such value is expressed in MIPS (i.e., millions of instructions per second) to consume, and it is also used to determine the VM lifetime. The more MIPS the hosting node can process, the less time the VM will last.

Thanks to virtualization, running a VM on different CPUs is not an issue. Conversely, accelerating the execution of a portion of the VM poses some challenges. Although for some kind of devices, it would theoretically be possible to calculate the equivalent number of MIPS performed since their functioning is driven by the execution of dedicated instructions, in case of FPGAs there is no instruction stream to execute. To overcome this limitation, we introduce a more general approach. We measure the number of instructions required by a given VM. Then, we measure the relative (to the specific hosting CPU) speed up using the accelerator, obtaining a *speed up factor* (F_{sp}). Thus, the number of MIPS associated to the host CPU (T_{cpu}) to consume the VM's MIPS is multiplied by F_{sp}. Such value expresses the throughput of an equivalent "accelerated" CPU (T_{acc}), as follows:

$$T_{acc} = T_{cpu} \cdot F_{sp} \tag{1}$$

While the power model of an accelerator such as a GPU or a many-core can be derived as a linear function of its loading factor, in case of FPGAs it is more complex. In fact, power consumption depends on the percentage of resources used, and the clock frequency of the synthesized design. To simplify, we analyzed a real application (i.e., convolutional neural network –CNN) and we estimated the power consumption for this design (on the Intel Arria10 midrange device), by fixing the clock frequency to a constant value of 170 MHz (we found that depending on the complexity of the synthesized circuit, the clock frequency span from 150 MHz to 350 MHz). This led us to generate the following model (R is the percentage of FPGA resources used by the design):

$$P_{A10} = 37.97 \cdot R + 27.5 \tag{2}$$

Equation 2 can be used to estimate the power consumption of other applications, by fixing the clock frequency to the same value while changing the amount of active resources on the chip. A similar model has been derived for the Intel Stratix10 high-end device (see also figure 3), as follows:

$$P_{S10} = 153.4 \cdot R + 110 \tag{3}$$

4 Simulations

Aiming at evaluating the effectiveness of the proposed allocation algorithm, we performed a large set of experiments using a State-of-the-Art simulation framework –*CloudSimPlus* [2, 3, 4], taking into account two different types of workload. Specifically, we described VMs requiring different numbers of MIPS to process (and cores), as well as different amount of memory. Virtual machines requiring a high number of MIPS to process have been marked for the execution on nodes equipped with an FPGA. The machine configurations in the simulated data center are heterogeneous too. We provided to the simulator machines equipped with processors exposing a variable number of cores and MIPS capacity. To support acceleration, some nodes have been marked as associated with an FPGA board. We considered two type of accelerators (midrange and high-end) to cover a broader spectrum of devices. The heterogeneity of VMs and hosting nodes allowed us to simulate workloads with higher dynamics (more typical for Cloud-oriented application), and workloads with long-lasting VMs (i.e, HPC-oriented workloads).

4.1 Experimental setup

Table 1 and table 2 respectively show the host configurations used for creating the data center during the simulations, and the types of VMs available to construct the workloads. We described both the hosting nodes and VMs in the CloudSimPlus simulation environment. The hosts are differentiated by means of the number of cores (and MIPS) exposed to the applications (ranging from 2 to 12 cores), and

the amount of main memory (ranging from 4 GiB to 12 GiB). For each node type, we provided two subversion: one equipped with an FPGA board, and one without accelerator. Whenever the FPGA is available, we indicated the speed up provided (when running the largest supported VM) by the accelerator. Finally, we generated a power model (based on equation 2 and equation 3) for the nodes, considering the minimum and maximum power consumption levels.

Aiming to evaluate the performance of ECRAE algorithms on different scenarios, we generated two workloads with different numbers of VMs to allocate, while keeping constant the number of hosts (i.e., 500 nodes). The first workload is characterized by a number of VMs that is twice the number of hosting nodes (i.e., 1000 VMs). The second workload aims at saturating the data center capacity, trying to allocate VMs that are $\times 4$ the number of available nodes (i.e., 2000 VMs). Network bandwidth and storage capacity was fixed for all the nodes, as well as for all the VM types. All the simulations have been performed on a Windows-7 (64bit) machine equipped with 8 GiB of main memory and an Intel Core i7 processor running at 2.5 GHz. CloudSimPlus requires the Java Runtime Environment (JRE) to correctly work, and in our experiments we used Java version 8.

Table 1: Host specifications

Host Config.	**MIPS**	**Cores**	**RAM**	F_{sp}	$\mathbf{P}_{min}$	$\mathbf{P}_{max}$
HP (Xeon 3075)	2660	2	4 GiB	$\times 1$	93.7 W	135 W
HP (Xeon 3075)	2660	2	4 GiB	$\times 10$	–	–
IBM (Xeon X5670)	2933	12	12 GiB	$\times 1$	66 W	247 W
IBM (Xeon X5670)	2933	12	12 GiB	$\times 30$	–	–
Intel Arria10	–	–	8 GiB	–	27.5 W	57.9 W
Intel Stratix10	–	–	32 GiB	–	110 W	225 W

Table 2: VMs specifications

VM type	**Total Instr.**	**Cores**	**RAM**	**Accel.**
1	12500	2	4096 MiB	true
2	2000	1	1740 MiB	false
3	10000	1	2100 MiB	true
4	500	1	613 MiB	false
5	1500	1	2000 MiB	false

4.2 Experimental results

ECRAE algorithm performance, i.e., the energy consumption of the data center running different workloads, have been compared with those provided by two largely adopted algorithms, i.e., First Fit (FF) and Best Fit (BF) algorithms. FF aims at providing the fastest decision, quickly allocating an incoming VM to the first node

with enough capacity. BF tries to better optimize the allocation by selecting the node having the highest capacity. Both the algorithms do not take into account power consumption of the nodes, thus leading to higher power(energy) usage when compared with ECRAE.

Simulation Results

Algorithm	#Hosts	Incoming VMs	Instantiated VMs	Energy consumed
First Fit	500	1000	1000	1425.69 kWh
Best Fit	500	1000	1000	1838.87 kWh
ECRAE	500	1000	1000	1006.33 kWh
First Fit	500	2000	1700	2050.39 kWh
Best Fit	500	2000	1700	2071.91 kWh
ECRAE	500	2000	1700	2030.03 kWh

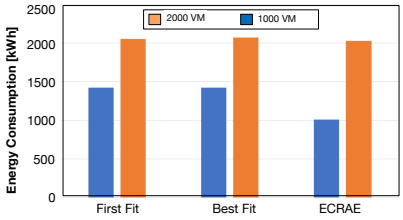

Fig. 4: Experimental results for the allocation of 1000 VMs and 2000 VMs on 500 hosts, using FF, BF, and ECRAE algorithms.

Figure 4 shows the efficiency achieved by the three allocation strategies on different workload conditions. We generated workloads according to the VM types listed in table 2; whenever VMs of type 1 and 3 must be allocated and no one of the machines equipped with an FPGA is available, then a free machine without acceleration support is chosen. This allows FF and BF to correctly search and allocate accelerated VMs on (a subset of) the available machines. From the results, it is clear that ECRAE is always able to take better decisions that minimize the overall energy consumption. On a medium load level (i.e., allocation of 1000 VMs), ECRAE provided 30% of energy saving over the FF solution, and 45.2% over the BF solution. This can be ascribed to the fact that, ECRAE searches a node, first, among running ones (avoiding machine in idle state, that were switched off). Conversely, BF search for the less loaded node for allocation, thus being induced (often) to select idle nodes. Since idle machines may drawn down up to 65% of peak power consumption, contribution of BF selected machines on the overall energy consumption was in most cases higher than in case of FF and ECRAE. However, FF is not driven by any consolidation-aware mechanism, thus globally selecting a worse mix of nodes if compared to ECRAE solution.

Trying to allocate 2000 VMs, caused the data center reaching the saturation point (i.e., actually not more than 1700 VMs were allocated). In this scenarios, all the allocation strategies should show a similar behavior. However, albeit less pronounced, ECRAE results were better than FF and BF solutions. Here, the energy saving was respectively of 1% and 2%. When compared to the pure performance, ECRAE should be able to access to accelerated nodes, thus in most cases providing better performance.

All these results, confirm the effectiveness of the proposed algorithm when compared to traditional and largely adopted strategies.

5 Conclusion

Heterogeneity, is growing in modern data centers to achieve better performance over always more heterogeneous workloads (e.g., HPC-oriented workloads, deep learning applications, etc.), as well as to lower the energy consumption of the infrastructures. Heterogeneity is primary obtained through specialized devices that can accelerate applications' execution on a specific domain. In order to correctly manage such vast computing power, smart power(energy)-aware allocation policies should be implemented in the orchestration tools. This paper presents ECRAE, an energy-aware VM allocation strategy. Being based on a greedy optimal selection criteria, ECRAE allows to achieve larger energy savings when compared to broadly adopted policies, i.e., First Fit (FF) and Best Fit (BF). Experimental results confirm the effectiveness of the ECRAE algorithm when workloads of different size and type are executed on the data center. Furthermore, ECRAE provides advantages over conventional algorithms, being able to exploit the availability of acceleration devices (e.g., FPGAs).

Acknowledgments

This work is supported by the OPERA project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the grant agreement No. 688386.

References

1. European H2020 OPERA project.
   http://www.operaproject.eu
2. Amazon Web Services (AWS) – Accelerated computing instances.
   https://aws.amazon.com/ec2/instance-types/
3. Microsoft Azure – GPU based instances.
   https://azure.microsoft.com/en-us/pricing/details/virtual-machines/series/
4. D. Foley, J. Danskin, "Ultra-Performance Pascal GPU and NVLink Interconnect", *IEEE Micro*, vol. 37, no. 2, 2017.
5. Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, Jeffrey S. Vetter, "NVIDIA Tensor Core Programmability, Performance & Precision", arXiv:1803.04014v1 [cs.DC], 2018.
6. A. Sodani, et al., "Knights Landing: Second-Generation Intel Xeon Phi Product", *IEEE Micro*, vol. 36, no. 2, 2016.
7. Benoit Dupont de Dinechin, et al., "A clustered manycore processor architecture for embedded and accelerated applications", *High Performance Extreme Computing Conference* (HPEC), IEEE, 2013.
8. Carl Ramey, "TILE-Gx100 ManyCore processor: Acceleration interfaces and architecture", *Hot Chips 23 Symposium* (HCS), IEEE, 2011.
9. Intel Stratix10 website.
   https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html
10. Intel Arria10 website.
    https://www.altera.com/products/fpga/arria-series/arria-10/overview.html

11. Microsoft project Catapult.
https://www.microsoft.com/en-us/research/project/project-catapult/
12. A. Putnam, et al., "A cloud-scale acceleration architecture", *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, IEEE/ACM, 2016.
13. A. Beloglazov, and R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers", *Concurrency and Computation: Practice and Experience* (CCPE), vol. 24, no. 13, pp 1397–1420, John Wiley & Sons, 2012.
14. M. C. Silva Filho, et al., "CloudSim Plus: a Cloud Computing Simulation Framework Pursuing Software Engineering Principles for Improved Modularity, Extensibility and Correctness," in *IFIP/IEEE International Symposium on Integrated Network Management*, 2017.
15. M. C. Silva Filho, et al., "CloudSim Plus: A Modern Java 8 Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services", https://github.com/manoelcampos/cloudsim-plus/blob/master/docs/cloudsim-plus-white-paper.pdf
16. R. N. Calheiros, R. Ranjan, et al., "CloudSim: a toolkit for modeling and simulation of cloudcomputing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, 41 (1) (2011), pp. 23–50
17. A. Beloglazov, J.Abawajy, R. Ranjan, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", *Future generation computer systems* (FGCS), 28 (5) (2012), pp. 755–768
18. A.H. El Bakely, H. A.Hefny, "Using shortest job first scheduling in greencloud computing", *Int J Adv Res Comput Commun Eng*, 4 (2015), pp. 348–354
19. A. Beloglazov, and R. Buyya, "Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints", *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no.7, 2013.
20. Bahwaireth, K., Tawalbeh, L., Benkhelifa, E. et al., *EURASIP J. on Info. Security*, (2016) 2016: 15. https://doi.org/10.1186/s13635-016-0039-y
21. E. Pinheiro, R. Bianchini, et al., "Load balancing and unbalancing for power and performancee in cluster-based systems", *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*, 2001, pp. 182–195.
22. Filelis-Papadopoulos, C.K., Giannoutakis, K.M., Gravvanis, G.A. et al., *J Supercomput*, (2018) 74: 530. https://doi.org/10.1007/s11227-017-2143-2
23. Casanova H., Legrand A., Yves R., "Parallel Algorithms", CRC Press, 2011.