Heterogeneous-ISA Application Migration in Edge Computing: Challenges, Techniques and Open Issues

Hang Jin[1], Yihong Su[1], Fengzhou Liang[1], and Fang Liu[1,2](✉)

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
{jinh26,suyh35,liangfzh}@mail2.sysu.edu.cn

[2] School of Design, Hunan University, Changsha 410082, China
fangl@hnu.edu.cn

Abstract. With the development of mobile edge computing, more and more services are moved to the edge of the network, and devices there are usually with low computational abilities and little storage resources. To make it lightweight and elastic, containers can be adopted in the edge environment when migrating a certain application. With the host OS kernel shared, applications can be deployed with the least computational resources they need, making it possible to deploy more of them on relatively low-end devices. Migration is also used in scenarios like maintenance or load balance, etc. We noticed that in edge environment, devices and servers are usually built with heterogeneous Instruction Set Architectures (ISAs) processors. X86 processors are widely used in desktop PCs, laptops and servers while smart-phones are built with an ARM processor, which leads to a serious problem that a container cannot be migrated to a heterogeneous machine to continue running directly. In this paper, we firstly give an overview of heterogeneous-ISA migration, and its applications and techniques. Then we discuss the existing heterogeneous execution solution from the perspective of applicable scenarios, latency, power consumption, requirements for computational resources, etc. Next, a comparison study is given on each of the characteristics to depict the details and differences in existing works. At last, challenges and open research issues which are waiting for further studies on container migration are listed.

Keywords: Heterogeneous migration · Edge computing · Container

1 Introduction

Edge Computing [1] is a novel computing paradigm in which computation are performed at the edge of the network. Data are processed closer to the user in both physical and in the topology of the network. Thanks for edge computing, instead of sending all data generated by mobile devices (e.g. smartphones and IoT devices) to the cloud, data can be processed at anywhere between the end user and the data center so the burden of backbone network bandwidth and data center is reduced and the request for a certain service can be responded faster than cloud computing.

X. Sun et al. (Eds.): ICAIS 2021, CCIS 1423, pp. 106–118, 2021.
https://doi.org/10.1007/978-3-030-78618-2_9

Virtualization technology is widely used nowadays to make full use of a physical server by dividing it into independent execution environments logically. The most commonly used virtualization technologies are Virtual Machines (VMs) and Containers. Normally, each VM represents a single full-system execution environment so it will provide user with a highly isolated environment by installing and running a full operating system inside which will occupies much computation and storage resources. In the age of cloud computing, VMs are deployed in data centers for isolation and they are transparent to users. With the development of edge computing, more and more relatively low-end devices, such as gateways, routers, wireless APs and mobile phones, are working at the edge of the network. Since these devices have little computation resources, it is not advisable to deploy VMs without a deeper consideration. In some circumstances, containers can be simply deployed and managed at the edge since they only require little computation and storage resources.

Migration is a technology commonly used by cloud service providers to migrate a VM from a certain physical server to another for maintenance or load balance propose in cloud computing while container migration is more likely to be used in edge computing to provide user with a certain service consistently though the mobile user is moving to a new place physically.

At the edge of the network, devices there are not always built in the same Instruction Set Architecture [2] which means that we cannot migrate a container built in a certain instruction set to another so heterogenous-ISA migration is proposed to provide a method for applications to migrate to any other edge nodes, regardless of what instruction set the source node and the destination node are.

However, after we studied the existing works which focus on heterogenous-ISA application migration, there is no public literature to make a comparative insight about this topic, especially container migration. In this paper, our research analyzes various solution in heterogenous-ISA migration problems and provide an overview of the existing challenges, techniques and open issue.

As we did our best to know, this paper is the first to provide a full overview of the heterogenous-ISA application migration, especially container migration. Contributions of this paper can be summarized as follows:

- Reviewing the solution of the existing works on application migration in mobile edge computing from the perspective of techniques.
- Analyzing the adaptive application scenarios and motivation of existing solution to migrate an application to a heterogenous-ISA platform.
- Helping to make a choice when selecting a solution or technique route to perform a heterogenous-ISA migration in production environment.
- Identifying the existing challenges and open issues of heterogenous-ISA migration which are worth more research conducted on.

The remaining part of this paper are organized as follows. In Sect. 2, we list and discuss several main solutions of running applications on heterogenous platforms. In Sect. 3, we classify this problem in detail from the perspective of different types of programming languages and their respective solution on this problem. In Sect. 4, we analyze some typical existing works from the perspective of main purposes. In Sect. 5, we make comparisons on the typical works, then list the challenges and open issues in heterogenous-ISA application migration. Finally, we summarize our conclusions in Sect. 6.

2 Virtualization and Migration

Execution isolation is essential for both cloud and edge applications, which can be realized by virtualization technology. We summarize this into measures below: simulation, VM and container. The rest of this section will discuss them in detail.

From the perspective of migration, there are two types of ways to migrate a VM or container: non-live migration and live migration. Non-live migration means the state of the runtime information is saved to a file for further use such as instance cloning, state backup and rolling back, etc. Live migration is used to provide consistent service, such as quick replication [3], load balancing [4], and keeping relatively closed to the user [2], etc.

2.1 Simulation

Simulation refers to migrating the application process to the destination platform and keeping running on a full-system simulator (e.g., QEMU [5]) on the destination site, and the simulator simulates the whole hardware of a computer. However, the simulated hardware is usually implemented by software which leads to a performance bottleneck compared with running on real hardware so it is not adaptable to be used in production environment in most circumstances.

2.2 Virtual Machine

Each virtual machine is an isolated execution environment on the host machine with its own guest operating system running inside. To deploy a VM, a hypervisor is necessary for management use. A virtual machine can be deployed right on the top of host hardware [6] or on the host operating system [7, 8]. Though VMs can provide relatively more independent execution environment for guest processes, the guest operating system installed will take much storages and memory spaces, regardless of whether a system component or service is utilized by the application.

In the age of cloud computing, VM is most popular virtualization solution utilized by data centers to divide an integrate server into small virtual machines logically while the hardware resources are shared physically. Cloud service providers (AWS, Aliyun, etc.) can also rent VMs to customers and provide them with a remote access to deploy their applications on these infrastructures, which is also known as Infrastructures as a Service (IaaS). As we described above, a virtual machine has a relatively isolated execution environment but it will occupy a great amount of physical resources.

VMs can be packed and exported to other physical servers by simply migrating the disk image and settings to achieve this. However, to migrate a VM lively, the whole state and data of the running VM should be all transferred to the destination site, including register state, memory, storage data and network connections. Existing method to lively migrate memory of a VM are summarized as pre-copy [9], post-copy [10] and hybrid-copy [11]. The storage data can be migrated by method similar to that of memory data migration. Though there is quite an amount of works on VM migration, it can be difficult to migrate them to a heterogenous-ISA platform since it is heavy and there are some OS-level limitation which is ISA-related. We will not make a further discussion about VM migration.

2.3 Containers

Container is a technology which is implemented by sharing the host OS kernel with the container's own file system. As containers can be generated and disposed easily and provide a relatively isolated lightweight execution environment for guest process, they are widely deployed in production for application packing, management and environment isolation. Since containers share the host OS kernel instead of executing processes in their own guest OS execution environment, the level of isolation is less than that of VMs.

Since containers are executed on the top of host OS, the migration of them are more similar to process migration. The usually used method of migrating a container/process lively is CRIU [12], an open source checkpointing tool for checkpointing and restoring an image of the migrated container/process. Similar with VM migration, container migration also consists of migration of register state, memory, storage data and network connections. In the rest of our paper, we will not differentiate container migration and process migration since they are basically similar to each other.

2.4 A Comparative Study

In this section, we summarize the advantages and disadvantages of the main ways of migration discussed above. Figure 1 shows the basic architecture of each specific measure of virtualization and Table 1 shows the comparative features of them.

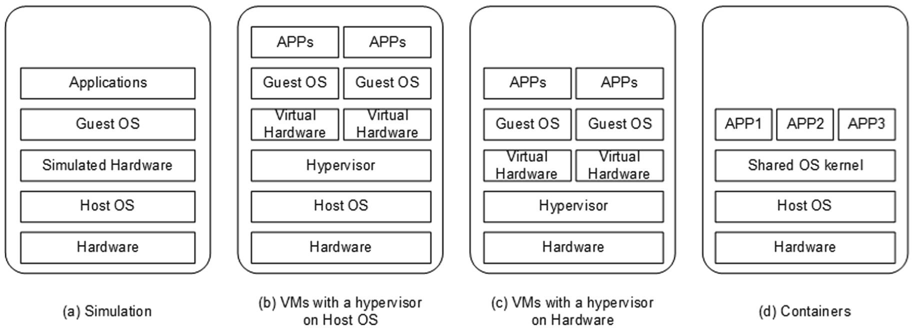

Fig. 1. Basic architecture of various measures of virtualization

Table 1. Features of various measures of migration

Method	Oriented scenarios	Performance	Volume of data	Migration velocity	Migration overhead	Support for heterogenous migration	Environment isolation
Simulation	Experimental	Very low	Large	Very slow	High	Fine	Fine
Virtual machines	Cloud	Normal	Large	Slow	Normal	Terrible	Fine
Containers	Cloud & edge	Normal	Little	Fast	Low	Acceptable	Acceptable

As we depicted above, simulation can lead to a serious performance bottleneck caused by its software simulation of electric hardware so it is more likely to be used in experiments rather than commercial use. There is a huge amount of works on VM migration, but that focuses on heterogenous migration can be rare since there are always an operating system running inside, which communicates with the virtual hardware and it's hard to migrate it to a heterogenous platform. As there are serval limitations for VM migration in the edge scenarios, containers can fit the edge computing environment better, since containers can be lightweight, and occupy less resources than VMs do.

3 Typical Works

In this section, we will discuss some typical works about container migration in detail. In the scope of container migration, applications nowadays are usually written in one or more high-level programming languages. These high-level programming languages can be classified as: interpreted languages (e.g. Java, Python, Perl, etc.), in which programs are written when going to be executed, an interpreter is indispensable to interpret the source code; compiled language (e.g. C, C++, etc.), in which programs are written should be complied by a compiler to generate an executable file which will be loaded by the OS when executed.

3.1 Interpreted Language Applications

For applications written in interpreted language, the runtime execution environment is usually provided and managed by the interpreter, such as a Java Virtual Machine (JVM) for Java programs and a Python Interpreter for Python codes. The runtime execution environment provided by the interpreter including I/O, system functions, Garbage Collection (GC), etc. which help the guest program run without being aware of what OS is it running on so it's relatively easy to migrate an application built on interpreted languages. As interpreted languages are usually provided with cross-platform interpreters, applications can run on the heterogenous-ISA platform [4] easily. Some optimization methods are also proposed to reduce the overhead of migrating an interpreted language application.

Chen et al. [13] proposed a programming framework COCA to offload computations in cloud computing. COCA utilized aspect-oriented programming (AOP) to offload Java applications in which computation can be offloaded to a destination site by inserting additional information into the application to be migrated.

Bruno and Ferreira [14] proposed ALMA, a solution for JVM live migration. The key idea of their work is to make a trade-off between the overhead of GC and the transmission of data that needed to be transferred to the destination site. ALMA check each heap memory periodically and estimate the time to be taken to collect each heap memory area. By comparing the estimated velocity to collect a certain memory area and the velocity of transmit this memory area to the destination site, the decision whether the garbage in this memory area should be collected to get a minimal time of the sum of the GC and transmission.

3.2 Compiled Language Applications

Compiled language applications can be loaded by the OS and executed directly and the code should be compiled to the instruction set of the target machine. So, when compared with interpreted language programs, it is much harder to make it compatible for a process to migrate across heterogenous platforms. There are some works proposed to extinct the boundaries of heterogenous-ISA platforms to migrate a process on a certain platform to another.

Barbalace et al. [4] focus on the energy consumption of a certain workload on an ARM and an x86 server. They implement a whole toolchain from the OS-level, including an extended OS based on Popcorn Linux [15], customized multi-ISA compilers to generate multi-ISA binaries and runtime support of migration to the heterogenous-ISA platform. The compilation toolchain in their proposed model finally link the program of heterogenous-ISA version of the source code, ensuring that symbol addresses of the program can be linked to the same address for alignment so that the related entry points of functions and variables can keep the same even running in heterogenous-ISA platforms for an easy state transformation.

Checkpoint/Restore In Userspace (CRIU) [12] is an open source project which can save the state of a container/process into an image file on the disk. And we can transfer the image files to the destination to restore the whole state of the migrated process. Note that since CRIU snaps the whole state of a certain process including code segment of the process, if the image were transferred to a heterogenous site with different instruction set, the process cannot be restored unless we exert extra operations to the image files. There are numerous works struggling to realize heterogenous-ISA migration based on CRIU.

Barbalace et al. proposed H-Container [2], which stands for Heterogenous-ISA Container, to migrate a container to the heterogenous-ISA platform. H-Container decompiles the executable of the migrated application into LLVM IR as the intermediate represent, and then insert some "migration points" for program to get paused at to trigger a migration operation. After this, the migrated application will be re-compiled to the destination instruction set. What's more, H-Container deals with the runtime state of the running application to transform the image file to fit the destination site then restore the execution of the container on the destination site.

Unikernel can be also used in cross-ISA migration. Oliver et al. proposed Heterogeneous EXecution Offloading (HEXO) [16], utilizing unikernel virtual machine which is an integrate OS-level application combined part of the kernel with user application to migrate workloads on embedded platforms to save energy in HPC data centers. HEXO requires migration points inserted into the application source code and compile it using a heterogeneous-ISA toolchain to generate images of multi-ISAs for offloading or migration.

Process migration has also been studied in heterogenous-ISA chip multiprocessors. DeVuyst et al. [17] studied how to migrate a process to a heterogenous-ISA core with little performance loss to make a full use of heterogeneous-ISA CMP. Their work includes identifying the program state, modifying the complier to compile a program with data properly placed in the memory when executed so that migration cost can be reduced and they also had a research on binary translation. The main idea of their work is to keep the form of a program running in the memory the same in both source core and the heterogenous destination core that the program can keep running natively to achieve an acceptable performance.

Bhat et al. [18] proposed an operating system with replicated kernels which is extended form Popcorn Linux and a customized compiler to migrate a process on an ARM-x86 heterogenous platform. Their work focuses on migration in the heterogenous platform which is composed of ARM and x86 architecture. There is a kernel on each architecture which is compiled natively and communicates with each other via Popcorn communication layer. Popcorn Single System Image runs on the both kernel and so does the application. The application is buillt with the customized compiler in which addresses of variables and function entry are placed to the same virtual address. The executable also contains codes of both ISAs, the OS will map the corresponding code in the executable to the same ISA. To migrate a process, the state on the source processor is going to be packed and sent to the destination ISA via the communication layer and restored after sending.

3.3 Hybrid Applications

Hybrid applications refers to applications that composed of one or more parts discussed above. Android, one of the most popular operating system for smartphones and smart SoCs, can be a great example for this. According to the official documentation [19], an Android application can run native binaries of libraries or modules that written in C codes for better performance. They are usually high-frequency or computation-intensive functions or codes, which is called NDK libraries and can boost the overall execution performance of the application.

Lee et al. [20] are the first to notice the migration of hybrid applications on Android platform. As they described, the most popular Android applications such as Firefox, VLC Player, etc., are built with C/C++ codes up to more than 50% of total codes to boost the performance of the APPs. Most of the smartphones with Android installed are equipped with ARM processors while servers and desktop PCs are usually x86. Noticing the performance gap of state-of-the-art smartphones and servers, this work proposed Native Offloader, to identify the heavy tasks of an application that are able to be migrated to a x86 server independently. The key of their work is the Native Offloader Compiler, which analyzes the native codes of the application, partitions the original code into client IR codes and server IR codes, then compiles IR codes for both platforms to enable migration. When the workload running, client submits related data including identifier, stack pointer and page table to the server and receives the dirty pages from the server via network connections.

4 Purpose of Migration

In this section, we studied the existing works and classify them from the perspective of main purpose and adaptable scenarios. We summaries this as several different aspects and we believe that it will be of great importance to get a better comprehension of these works.

4.1 Following User Mobility

Edge computing indicates offloading computation closer to the user to provide user with low-latency services, and this will leads to a problem of following the user. In this context, users are with great mobility, and so are the IoT devices.

The main purpose of H-Container [2] is to utilize as more potential target in the edge cloud as possible to migrate those services in need of low latency such as gaming and real-time calculation. There is a strong heterogeneity at the edge of the network that edge devices can be servers, PCs, laptops or any embedded SoC devices such as routers, gateways or wireless APs. These devices are built with x86, ARM and other CPU processors. By enabling migrating containers to heterogenous-ISA platforms, there can be more candidate places to migrate a container that is serving the current user onto. So the possibility to migrate an application to closest edge device to provide the user with a consistent low-latency service when the user is moving to another edge cloud is boosted.

4.2 Performance

The weak ability of computation can also be one of the serious problems that IoT devices at the edge of the network have, since these embedded devices are usually equipped with relatively low performance processors. When users are running some computation-intensive applications on their smartphone, they will be willing to migrate their workloads to an edge device with stronger computation ability to boost the performance of the application. This idea can be found in [20], which migrates partitioned computation-intensive code from heavy tasks of an application to the server to acquire better performance.

4.3 Energy Efficiency

Energy efficiency is now attracting more and more global attention to achieve the purpose of environment protection. And data centers are also trying to do so to cut down the electricity cost. Experimental evaluation of [4] indicates an average of 30% energy reduction by migrating applications to ARM platforms. The similar purpose can be found in HEXO [16] for HPC workloads migrating to embedded devices with the unikernel virtual machine discussed above. Works of [17, 18] are also intended to get a greater energy efficiency by migrating processes onto heterogenous-ISA multiprocessors. The migration studied in this work is about the migration between processor cores instead of that between heterogenous machines.

5 Discussions

In this section, we will give some discussions on the works we mentioned above, summarize the features of typical works and give an overview of them. Then we discuss how a decision can be made among these valuable works from the perspective of applicable scenarios, purpose and techniques, when a migration is required. At last, the existing challenges and open research issues in heterogenous-ISA application migrations are listed.

5.1 Comparative Study

We summarize the scenarios, techniques and advantages of some works discussed above, which are listed in Table 2. The timeline of these works is depicted in Fig. 2, which gives an explicit relation and the development of them.

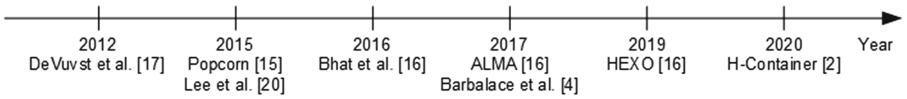

Fig. 2. Timeline that typical works are proposed

Table 2. Comparison of existing typical works

Aspect	DeVuyst et al. [17]	Lee et al. [20]	Bhat et al. [18]	ALMA [14]	Barbalace et al. [4]	HEXO [16]	H-Container [2]
Virtualization Technology	Container	None	None	JVM	Container	Unikernel	Container
Year	2012	2015	2016	2017	2017	2019	2020
Platform	Heterogeneous multicore	Mobile	Heterogeneous multicore	Universal	Universal	Universal	Universal
Workload	Compiled	Hybrid	Compiled	Interpreted, Java APPs	Compiled	OS-level Application	Compiled
Scenario	Heterogeneous multicore	Native Android code	Heterogeneous multicore	Universal	Universal	HPC data center	Edge
Main purpose	EE, PF	EE, PF	EE	FT, LB, EE, FS, LU, etc	EE	EE, PF	UM
System component	CP, BT, ES, SP	CR, IR	CP, ES, SP	JVM, CRIU	CP, IR	CP, Unikernel	CP, IR, BT, CRIU
Limitation	Rare platform in edge environment	Lack of demonstration for real Android Apps	Rare platform in edge environment	For Java applications only	Source code is required	Not in real edge environment	Lack of demonstration for security
Mobile Device Support	No	Yes	No	No	Yes	No	Yes

Sorted by year the work is proposed from left to right. The abbreviation list: FT: Fault Tolerance. LB: Load Balance. EE: Energy Efficiency. FS: Fast Start. LU: Live Updates. UM: User Mobility. PF: PerFormance. CP: ComPiler. ES: Extended System. SP: Special Platform. IR: Intermediate Represent. BT: Binary Translation.

5.2 Scenarios and Selection

In this section, we will summarize typical works from the other view of user choice. Figure 3 shows the process to select one of existing techniques and solutions, which will help user decide when choosing to apply one of these solutions in the future. Note that the final decision is not always fitting to a requirement perfectly and some of that can be developed from the nearest solution, e.g. when migrating a Python application, it might be derived from ALMA [16] or other similar works which are not listed since Python applications are running in a similar way that Java applications do while both of the languages are interpreted ones.

5.3 Open Research Issues

Though some typical works about application migration are proposed, there are still some open issues remaining to be addressed. With the development of edge computing, there will be more importance and challenges in the field of application migration.

User Privacy and Migration Security. Privacy protection is now drawing more and more attention for mobile users and enterprises. As an important measure to migrate an application, more attention should be paid on the privacy and security protection. An application cannot run without data to be processed. In the circumstance of edge computing, when users are enjoying the convenience brought by application migration, how

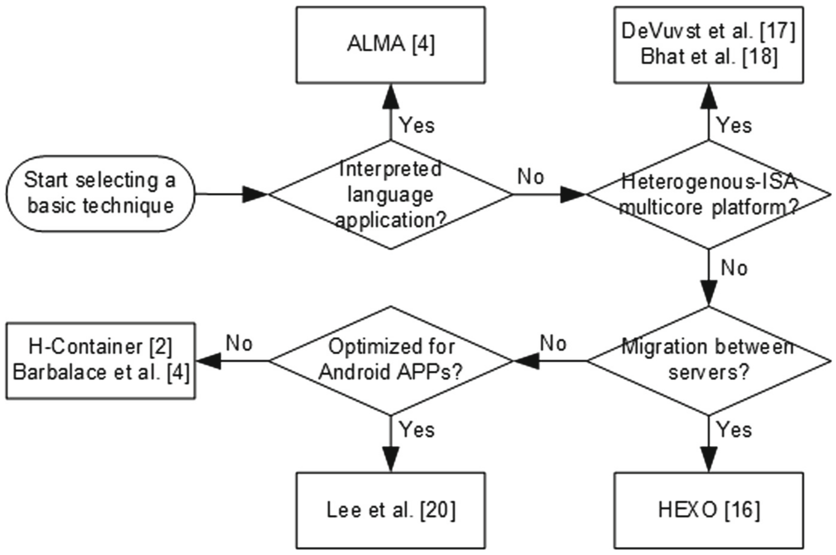
Start selecting a basic technique
Interpreted language application?
Yes
ALMA [4]
No
Heterogenous-ISA multicore platform?
Yes
DeVuvst et al. [17]
Bhat et al. [18]
No
Migration between servers?
Yes
HEXO [16]
No
Optimized for Android APPs?
Yes
Lee et al. [20]
No
H-Container [2]
Barbalace et al. [4]

Fig. 3. A decision-making guide diagram

to ensure the safety of data generated or provided by users can be an essential problem to be studied. The protection includes not only the safety of VMs or containers user application running in, but also the communication between the source and destination site when executing a migration operation. More effective measures should be proposed to boost the security of the data and user privacy.

Flexible Heterogenous-ISA Application Migration. As typical works show, when applying a heterogenous-ISA migration solution, there are always modification to be applied to the source code of the application, which is not friendly for developers to develop or update their applications.

Optimization for Particular Edge Scenarios. Although techniques of heterogenous-ISA migration have been studied in various platforms and they are likely to be adopted by edge scenarios, there are still little works focus on real scenarios in edge computing, especially for users with smart devices. To adapt to the edge computing scenario, solutions should be optimized for the real edge environment and evaluated, while existing works commonly focus on the performance of proposed solution itself.

6 Conclusion

In this paper, we described some typical works and solutions dedicating to solve the problem of heterogenous-ISA application migration. They are all likely to be adopted to boost the edge computing for purpose of quality of user services, performance and energy saving, etc. We also give a comparative study on the existing techniques to summarize their applicable scenarios and benefits by applying corresponding solutions.

Then we presented a guidance for decisions on selecting migration solutions. At last, open research issues are listed.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (62072465).

References

1. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
2. Barbalace, A., Karaoui, M.L., Wang, W., Xing, T., Olivier, P., Ravindran, B.: Edge computing: the case for heterogeneous-ISA container migration. In: 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE) (2020)
3. Lertsinsrubtavee, A., Ali, A., Molina-Jimenez, C., Sathiaseelan, A., Crowcroft, J.: PiCasso: a lightweight edge computing platform. In: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet) (2017)
4. Barbalace, A., et al.: Breaking the boundaries in heterogeneous-ISA datacenters. In: Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (2017)
5. QEMU. https://www.qemu.org/. Accessed 23 Dec 2020
6. What is ESXI? Bare Metal Hypervisor, ESX, VMware. https://www.vmware.com/products/esxi-and-esx.html. Accessed 23 Dec 2020
7. Oracle VM VirtualBox. https://www.virtualbox.org/. Accessed 23 Dec 2020
8. Windows VM, Workstation Pro, VMware. https://www.vmware.com/products/workstation-pro.html. Accessed 23 Dec 2020
9. Clark, C., et al.: Live migration of virtual machines. In: 2nd Symposium on Networked Systems Design and Implementation (NSDI) (2005)
10. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. ACM SIGOPS Oper. Syst. Rev. **43**(3), 14–26 (2009)
11. Hu, L., Zhao, J., Xu, G., Ding, Y., Chu, J.: HMDC: live virtual machine migration based on hybrid memory copy and delta compression. Appl. Math **7**(2L), 639–646 (2013)
12. CRIU. https://www.criu.org/Main_Page. Accessed 23 Dec 2020
13. Chen, H., Lin, Y., Cheng, C.: COCA: computation offload to clouds using AOP. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (2012)
14. Bruno, R., Ferreira, P.: ALMA: GC-assisted JVM live migration for java server applications. In: 17th International Middleware Conference (2016)
15. Barbalace, A., et al.: Popcorn: bridging the programmability gap in heterogeneous-ISA platforms. In: Tenth European Conference on Computer Systems (EuroSys) (2015)
16. Olivier, P., Mehrab, A.K.M.F., Lankes, S., Karaoui, M.L., Lyerly, R., Ravindran, B.: HEXO: offloading HPC compute-intensive workloads on low-cost, low-power embedded systems. In: 28th International Symposium on High-Performance Parallel and Distributed Computing (HPDC) (2019)
17. DeVuyst, M., Venkat, A., Tullsen, D.M.: Execution migration in a heterogeneous-ISA chip multiprocessor. In: The seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (2012)
18. Bhat, S.K., Saya, A., Rawat, H.K., Barbalace, A., Ravindran, B.: Harnessing energy efficiency of heterogeneous-ISA platforms. ACM SIGOPS Oper. Syst. Rev. **49**(2), 65–69 (2016)

19. Concepts, Android NDK, Android Developers. https://developer.android.com/ndk/guides/concepts. Accessed 23 Dec 2020
20. Lee, G., Park, H., Heo, S., Chang, K.A., Lee, H., Kim, H.: Architecture-aware automatic computation offload for native applications. In: 48th International Symposium on Microarchitecture (MICRO) (2015)