# Live Migration on ARM-based Micro-datacentres

Ilias Avramidis*, Michael Mackay*, Fung Po Tso†, Takaaki Fukai‡, Takahiro Shinagawa§
*Department of Computer Science, Liverpool John Moores University, UK
†Department of Computer Science, Loughborough University, UK
‡Department of Computer Science, University of Tsukuba, Japan
§Information Technology Center, The University of Tokyo, Japan
Email: {i.avramidis@2015.ljmu, m.i.mackay@ljmu, p.tso@lboro}.ac.uk; {fukai@osss.cs.tsukuba, shina@ecc.u-tokyo}.ac.jp

***Abstract*—Live migration, underpinned by virtualisation technologies, has enabled improved manageability and fault tolerance for servers. However, virtualised server infrastructures suffer from significant processing overheads, system inconsistencies, security issues and unpredictable performance which makes them unsuitable for low-power and resource-constraint computing devices that processing latency-sensitive, "Big-data"-type data. Consequently, we ask: "How do we eliminate the overhead of virtualisation whilst still retaining its benefits?"**

**Motivated by this question, we investigate a practical approach for a bare-metal live migration scheme for ARM-based instances low-power servers and edge devices. In this paper, we position ARM-based bare-metal live migration as a technique that will underpin the efficiency on edge-computing and on Micro-datacentres. We also introduce our early work on identifying three key technical challenges and discuss their solutions.**

***Keywords*—Edge Computing, Micro-datacentres, ARM Virtualisation, Hypervisor, Live Migration, Bare metal Clouds.**

## I. Introduction

For many years enterprise administrators have adopted Cloud Computing paradigm and technologies as an efficient alternative for hosting data, services and applications instead of owning, managing and maintaining their own private data centre infrastructures. Despite the broad utilization of Cloud Computing, surveys have long shown that virtualised cloud data centre infrastructures are unsuitable platforms for hosting high-performance computing, latency-sensitive applications or services due to the perceived lack of performance, unacceptable latency, location-awareness and security issues [8, 11]. Moreover, the rapid growth of IoT (Internet of Things) devices connected to a network generates massive amounts of data and information that need to be processed at highly centralized resources on cloud data centres for storage and computation, dramatically increasing the network latency. In addition, as the physical distance between edge users and the cloud increases, network transmission and response times increase too. As such, many application and services like "Big-data"-type, e-commerce, live-streaming, real-time online gaming cannot benefit from this paradigm.

In order to address these shortcomings, Cloud Service Providers (CSPs) have begun to change the centralized data centre model to a distributed, modular, decentralized architecture by introducing edge computing solutions. Edge computing is the shift of moving processing power nearer to the edge of a network and closer to the user, minimizing the network latency to maintain the performance of storage, processing and computation time while improving the transmission and response times between an edge user and Cloud [14, 18]. This is achieved through the implementation of Micro-modular Data Centres (Micro-DCs) which are containerised resource-rich systems located at the edge of a network, collecting, analyzing and processing the data generated by a variety of IoT devices.

As the same time, the hardware infrastructure of data centres is changing to maximise energy efficiency and optimise processing for 'Big Data'-type applications. The emergence of the new platforms called Bare-metal Clouds, aims to provide high levels of native hardware performance by eliminating the virtualisation layer whilst also offering better privacy and security. Furthermore, in order to address these changing requirements in the context of the move towards containerization and micro-datacentre infrastructures, we have started to see a move away from traditional architectures and the emergence of new vendors such as ARM on the server market. Newer generations of ARM architectures now have increased support for virtualization and offer competitive performance while minimizing costs and energy consumption. This combination of ARM CPU architectures and micro-modular data centres can provide a compelling combination of low cost, energy-efficient and scalable processing with minimal overheads and far greater service flexibility.

However, reducing the potential for a single point of failure and supporting serivce mobility are two of the main concerns of any administrator. Live migration therefore remains a valuable tool for infrastructure management, affecting a range of processes such as availability and accessibility, redundancy, fault tolerance, proactive maintenance and load balancing. Also, edge users now need the support for location-independence whilst moving from one edge network to another without affecting the connectivity and accessibility to applications, services and resources.

This paper proposes the development of a practical approach of a live migration process on ARM-based bare-metal micro-datacentres, to migrate physical machine states without the need for a virtualization layer. Such a platform would reinstate many critical management features between and for edge-networks and act as a catalyst for the deployment of a new range of edge computing. Although, over the years, a lot of different techniques and approaches have been proposed around native live migration only a few are applicable to bare-metal instances. However, this also poses a great number of challenging research questions to be addressed including, how to effectively capture the state of a running ARM server including the CPU, memory, and connected peripherals, and how to dynamically transfer and configure these in a new host to effect seamless migration.

The rest of this paper is organised as follows. Section 2 explores Micro-datacentres and bare-metal instances in more detail and Section 3 presents the Live migration process on micro-datacentres. In Section 4 we present our research proposal and explore the technical challenges involved. Finally, we conclude in Section 5.

## II. Micro-datacentres and Bare-metal instances

With the rise of the IoT, the number of IP-connected devices on a network is expected to continue increasing rapidly. However, many of these will be strictly attached to services and applications hosted on remote centralised locations like Clouds. IoT devices generate a massive volume and variety of data that must be processed and responded to in a very short time. However, this data at first need to cross a number of unknown gateways and other network devices until reach a data centre for storage, analyzing, processing and computation before responding back to the source. As a consequence, network bandwidth may be exhausted while the transmission and response times increase latency and service bottlenecks [14, 18].

The edge computing paradigm aims to change the trend of data-consolidation into one centralised data centre, by replacing this centralised client/server communication model with a decentralised, modular architecture by introducing Micro-Data Centres. Edge computing brings the data centre processing power nearer to the source of the data, and closer to the edge consumer, saving time and network resources while speeding up the processing and computation performance of data. Micro-DCs are modular, containerized and much smaller in size than traditional data centres and are typically located at the edge of each network in crucial positions such as in Base Stations or in apartment buildings. Each of them is integrated with storage, networking, compute, power resources, generally in a density of a rack-type size as illustrated on the left of Figure 1. Micro-DCs designed to solve, handle and processing specific type of data, important to the edge user. Thanks to their modular design reduce power-consumption costs while offer the characteristics of flexibility and mobility [14, 18].

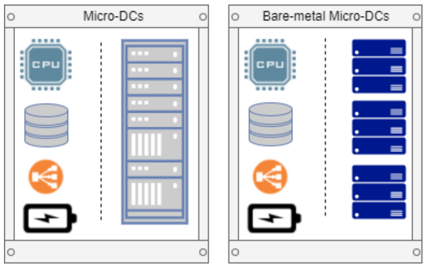

Figure 1: Micro-DC and Bare-metal micro-DC

Virtualisation Technology plays a key role in edge-computing, and especially in data centre management, by reducing the cost overheads while optimizing utilisation through maximising the capital investment in hardware resources, reducing energy consumption and space floor requirements, and a variety of consolidation schemes [8]. Even in Micro-DCs, virtualisation retains a core functionality to enhance the essential characteristics of scalability, elasticity and flexibility. However, virtualised server infrastructures suffer by significant performance degradation and security issues. Two of the most common performance issues that we see in virtualised environments are the processing overhead called "*Hypervisor tax*" and the symptom of "*Oversubscription*" [8, 9]. Virtualized servers are multi-tenant platforms, as a consequence, during intensive operations edge users may have to compete with each other for access to hardware resources, whilst all customers are served through the same physical network interface card (NIC) [7]. Consequently, all the available bandwidth is shared among them, potentially causing bottlenecks, delays and network inconsistency as well increasing the security risk. Moreover, in a micro-datacentre paradigm, servers have to analyze, compute and process a wide variety and high volume of data from a large number of sources.

These issues led to the development of a new infrastructure model, called Bare-metal Clouds, which aims to provide native hardware performance capabilities and security whilst offering better management by grouping or categorising the computation process of the data, services and applications in more efficient way. [9]. Bare-metal Clouds as illustrated in Figure 2 are simply server boxes that do not utilize any form of Virtualisation Technology but are still managed and provisioned in a similar way. Bare-metal Clouds are becoming an attractive platform for services and applications who demand extreme and predictable processing performance, stability, network consistency, privacy and security. For example, applications that have to manage and process massive amounts of data in a short period of time and to handle I/O-intensive requests like, Big-data DBMS (Database Management System) applications, HPC (High Performance Computing) applications, E-commerce, live streaming etc. are natural candidates for this [9]. They are therefore a good potential fit for Micro-DCs, as illustrated in the right side of Figure 1.

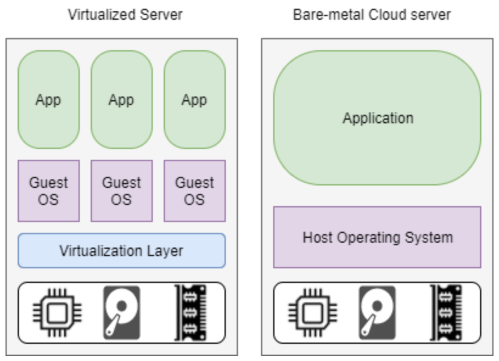

Figure 2: Virtualized and Bare-metal Servers

## III. Live migration on micro-datacentres

One of the key characteristics that makes Server virtualisation so attractive is that VMs are highly portable and administrators have the ability to transfer VMs between servers, racks even among data centres located in different geographical locations to increase the efficiency and effectiveness of virtualised server infrastructures. This is a powerful tool that enhances fault tolerance whilst enabling a number of proactive management and maintenance activities increasing the availability, accessibility and efficiency of services. Moreover, servers typically reach up to 70% of power consumption rates, even at low levels of utilization, greatly increasing the total expense for operators even for micro-datacentres. Live migration permits *Server consolidation* by hosting the same number of Virtual Machines in fewer physical servers and putting others into an idle state, as illustrated in Figure 3 [6].

Conversely, the live migration process also allows operators to maintain higher levels of Quality of Service (QoS) even

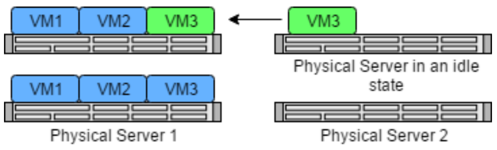

Figure 3: VM Server Consolidation

during peak periods of load. Over-utilization of servers leads to performance issues like congestion, latency and bottlenecks which can in turn create delays or interruption of services. In response, live migration can perform *load balancing* by sharing the workload from an over-utilized host to an under-utilized host as shown in Figure 4 [6].

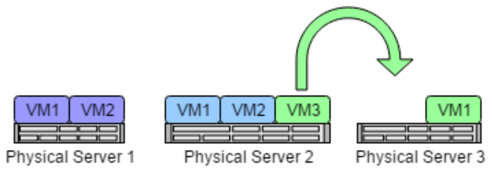

Figure 4: VM QoS Live Migration

Micro-datacentre infrastructures change radically when we consider bare-metal instances instead of virtualised server machines. The density of the physical servers has to remain low, into rack-size limits, which increase the importance of a live migration tool between bare-metal instances, which poses significant challenges for efficiency, scalability, redundancy and fault tolerance. Furthermore, the consolidation and load balancing techniques that take place in virtualised environments is much more challenging as administrators are faced with the task of performing live migration not only in the levels of a single micro-datacentre but also among micro-datacentres themselves as illustrates in the Figure 5. As already happens in Cloud data centre infrastructures, live migration must be seamlessly performed when edge users are moving from one micro datacentre to another without affecting or loosing their connectivity.

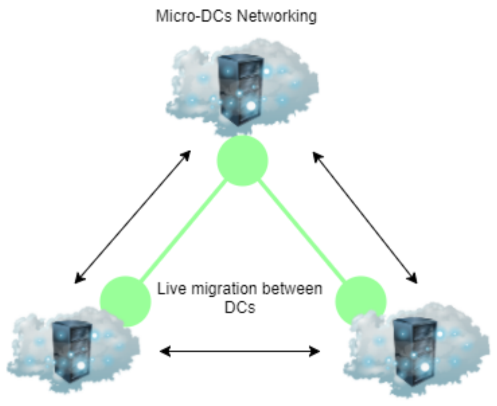

Figure 5: Live migration between Micro-datacentres

However, assuming that micro-datacentres are deployed with homogeneous hardware among bare-metal instances, we can envision a scenario whereby it is possible to capture, transfer, and redeploy the state of the underlying hardware between instances in the same way as is done with live migration. This approach, called Bare-metal Live Migration (BMLM), restores many of the key features of virtualisation including maximizing the power utilization of a micro datacentre to consolidate and power off unused bare-metal instances. However, while the functionality to achieve and perform this on Bare-metal server infrastructures is highly desirable, a live migration tool to do this has not yet been fully developed to the best of our knowledge.

The implementation of a live migration process on bare-metal instances is not straightforward because one has to deal with the states of the physical hardware resources and network connections that have to been migrated from one physical server to another. In bare-metal micro-DCs, to achieve the continuity of a process or application we have to ensure that the hardware states and connections of CPU, memory, storage, network and peripheral devices are kept in place and unaltered. The best way to achieve this is currently an active research issue.

As outlined above, an additional requirement of enabling BMLM is for the hosts to be suitably small, cheap, and power-efficient such that they can be deployed at scale in bare-metal micro-datacentres. The adoption of the ARM chip architecture therefore becomes a great potential solution to this issue.

## A. ARM Architecture

For many years, Cloud Computing vendors have tried to implement new ways to save energy and minimize the overall power consumption rates in data centre infrastructures, and the x86 architecture has dominated this space being almost ubiquitously deployed in the Desktop and Server enterprise world [15]. However, while big improvements in power management have been made over the years, these chips are fundamentally over-provisioned for future Micro-DC Bare-metal Cloud hosts. To address this, ARM have introduced its Cortex processor family to benefit by their small size, low-cost of deployment, and power-efficiency, making them an attractive alternative for these data centre infrastructures. [17]. Providers like Rackspace and Google have already expressed an interest in adopting ARM-based servers into their data centre premises as part of their future plans. ARM works in a licensed manner, meaning that chip vendors have the ability to buy a license for an ARM architecture set and then develop, design and deploy their own custom cores based on the instruction set. Chip vendors therefore have the advantage to design and adjust custom processors for specific applications making them very flexible. Qualcomm and Broadcom [15] are two of the biggest ARM designers and have already announced the production of 24 core ARMv8 CPUs to meet the performance requirements of the data centre industry. Moreover, the latest generation of ARM processors are further enhanced with Virtualisation Extension capabilities (Cortex-A17 model and later) and improved Security features [15, 17].

ARM introduced virtualization support extensions to its architecture in the ARMv7 and latest ARMv8 processors. A fundamental part of these extensions is the introduction of a brand new exception layer (EL2) as well as a CPU mode called Hyp mode. A hypervisor running in Hyp mode has higher privileges than the kernel (EL1) and application (EL0) modes and so can control OS access to the underlying hardware. Furthermore, the ARM architecture introduced a System Memory Management Unit (SMMU) and a 2-stage page translation process handling the translation and mapping of a virtual address (VA) to a physical address (PA) via a translation page table. In the first stage, the Virtual Address (VA) is mapped to an Intermediate Physical Address (IPA) which is controlled by the OS while at the second stage, the hypervisor performs the translation of the IPA to the Physical Address (PA) [5, 17, 12].

Although, the adoption of ARM-based servers show great potential for bare-metal Clouds, a practical solution and implementation of the bare-metal live migration process is still needed. By leveraging the new features outlined above, our aim is to develop a BMLM solution for ARM-based cloud servers.

## IV. Related Work

Many different approaches for live migration have been proposed by the research community which culminated with the OS-layer live migration process through Virtualisation architectures that we see today. As this is not suitable for Bare-metal Clouds, we therefore revisit research that focuses around the concept of Physical Machine hardware state Live Migration (PMLM) by omitting the virtualization layer. The PMLM process is still a challenging concept to the research community with a lot of issues to overcome.

Chiang et al.[4, 3] have proposed two prototype schemes for hardware state live migration, called PMSM (Physical Machine State Migration) and its successor, BOMLO (BOotstrapped Migration for Linux OS). Both schemes are based on the *hibernation* technology which is common in modern OSs like Linux and Windows. This enables taking a snapshot of the hardware state and keeping them on the system hard disk whilst the machine is powered off. When it powers up again, it automatically reloads the OS and stored image into memory and restores the system to its previous state. Their claim is that by migrating the snapshot image from one machine to another, the destination could resume execution of the system at the same state that the first one was suspended in. However, until now this approach does not have a practical and efficient implementation. Their practical experiments are based on the Linux hibernation feature called *TuxOnIce*. However, PMSM as well BOMLO do not migrate I/O device states which is critical for full machine migration [4, 3, 1].

Another research approach to PMLM was introduced by Fukai et. al. [13] which, to our knowledge, is the only effective implementation that also finds practical application on bare-metal instances. Their approach is based on the development and implementation of a thin, novel hypervisor layer named *BitVisor* [2]. The BitVisor is a bare-metal software based on x86 architecture and supports many OSs like Linux-based kernels, Windows and Mac. As Fukai's introduced on his project, by taking advantage of Intel's CPUs Virtualisation Technology extensions features, can achieve live migration of source CPU and memory as well as network and other I/O peripheral device states. BitVisor runs and is executed directly on the hardware prior to the OS booting, making it OS-independent. It obtains and sets CPU states by leveraging the hardware-assisted virtualization features that Intel virtualisation offers, whilst memory states can be captured and set by reading and writing them into the memory address directly. The most challenging part that BitVisor accomplishes is the migration of the peripheral devices states like network interface cards.

## V. Live Migration on ARM-based bare-metal MDCs

### A. Proposed Approach

By leveraging the advantages of modern ARMv8-A model processors, our research goal is the development of a lightweight hypervisor layer running directly on ARM architectures to support live migration of Micro-DC bare-metal instances. This will be a significant step forward because, as described in section 3, ARM-based servers are a likely candidate for bare-metal edge computing and most existing hypervisors are too heavyweight and not suitable to be applied on this hardware. The main goal of our hypervisor is to perform BMLM without the need for isolation and complicated management operations since we support only a single user per host, making it much lighter.

Under normal operation of the server, the hypervisor is transparent to the user, waiting for the live migration process to be triggered without affecting the performance of the system. By using a network boot protocol like PXE, the OS is booted from shared network storage accessible via a network protocol like iSCSI or NAS. The functionality of the hypervisor is therefore independent of the OS, meaning that it does not need the cooperation of the OS to capture hardware states of the system. Our live migration scheme also therefore does not need to consider storage migration, similar to a typical VM Live migration where the content is maintained on a shared device. During the Live migration process, first the hypervisor suspends the OS and obtains all the physical machine state, sending them to the hypervisor running on the destination machine. Then, the hypervisor on the destination machine will apply the state to the hardware, reconnect, and resume execution of the OS.

### B. Challenges & Potential solutions

The Live migration process on bare-metal hosts is very challenging since migration of physical hardware states needs to be performed. A VM image is made up of the states of CPU, memory, network and I/O peripherals device connections, which are encapsulated in a file that is transferred when a live migration takes place. In the same manner, but with a complete different methodology, in order to achieve this we need to determine how to capture and restore the states of CPU, memory and I/O devices included networking connections. Although Fukai et. al.[13] have already accomplished live migration on bare-metal Clouds using the x86 architecture, to our knowledge there is no similar work for ARM architectures. The main challenges that we have to overcome are therefore how to capture and set CPU states using the ARMv8 architecture, handle the migration of memory contents based on a *pre-copy* method [6], and capture and set the network and I/O device states.

#### 1) Migrating CPU states

The hardware-assisted virtualization features that most modern processors are enhanced with, came to improve the efficiency of software virtualization operations that hypervisors like KVM, XEN, and ESXi offer. Intel first introduced a series of Hardware Virtualisation Extensions (HVEs) consisting of VT-x, VT-d and VT-c, offering virtualisation of CPU, memory, I/O devices and networking infrastructure respectively. AMD likewise soon introduced its hardware virtualisation features (AMD-V and AMD-Vi) offering similar functionality but, despite their similarities, there is no compatibility between those two architectures [16]. On Intel, when a processor enables hardware virtualization it is entered into VMX operation. In general, a hypervisor runs on a root operation whilst the guest OS and software run on a non-root operation. The hypervisor is responsible for managing transitions between guest VMs and the host environment as well as handling interrupts that need an operation to be taken by the hypervisor. These transitions are controlled by a data structure called the Virtual Machine Control Structure (VMCS) which is accessed through a pointer. A VMCS is maintained per logical processor per virtual machine which has a region in memory allocated to a VMCS to keep track of the host and guest processor states [16]. So, by reading the host CPU states from the VMCS of the source machine and writing them to the VMCS at the destination machine, we can achieve CPU state migration.

However, this kind of structure is not supported by ARM. On ARM, Virtualization Extension capabilities were first introduced on ARMv7 generation processors (Cortex-A15 and later) and were enhanced and improved in the latest generation of ARM architecture (ARMv8), giving the ability to run multiple unmodified Guest OSs. As discussed above, ARMv8 introduced a brand new privilege layer (EL2) as well a new CPU mode called Hyp mode, as illustrated in Figure 6. A software running in Hyp mode has higher privileges than kernel mode which is running in EL1 exception layer. That permits us to install a hypervisor at EL2 layer controlling with less complexity the communication of the underlying hardware infrastructure and a Guest OS environment. Since ARM hardware-assisted virtualisation features do not support a VMCS memory structure, they do not automatically retain all guest and host processor states. Instead, ARM architectures enable the flexibility for software running on the EL2 exception level to decide which processor state it wants to save [12, 10].

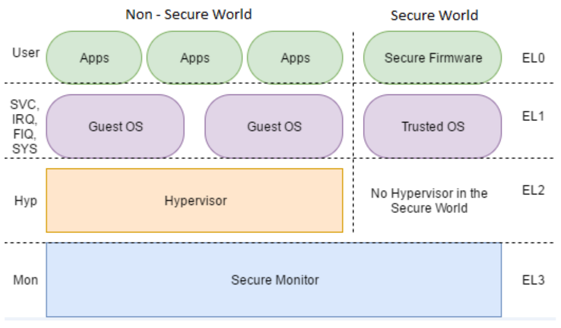

Figure 6: ARMv8 Exception Levels and CPU modes

On the ARMv8 architecture, interrupts and exceptions have the same meaning and are used alternately. When an exception occurs at a specific Exception level (ELn), the processor saves and stores the current execution state as well as the return address to that level. Each Exception level has its own bank of registers giving flexibility to a software to obtain the state at any time. This exception return state is held in dedicated registers for each of the Exception levels except the EL0, ELR and SPSR registers. As Figure 7 describes, the ELR register contains the return address where the interrupt occurred whilst the SPSR register holds the processor state. There is a bit in this register called IL that keeps the PSTATE value which contains the processor state before taking an exception and is used to restore the value of PSTATE when executing an exception return. We propose that software running on the EL2 Exception layer will be able to perform a CPU migration process similar to using VMCS functionality by obtaining and migrating the states and contents of those two registers (ELR, SPSR) [12, 10].

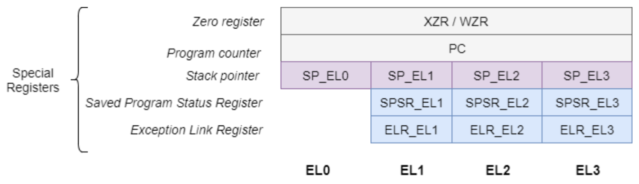

Figure 7: ARMv8 - A model special registers

### 2) *Migrating memory data*

Migration of memory states from a source to a destination machine is one of the most crucial components of a live migration process. The amount of memory contents is far higher than CPU state with capacities of up to 16GB of sensitive and volatile data being typical in most server systems. Therefore, the efficient and consistent migration of memory contents is very important where applications and services should continue to run without interruption. However, not all the assigned memory of a VM is used in practise so current techniques try to achieve better performance by transferring only the used contents of memory, eliminating the downtime as well as the total migration time. In general, the live migration of memory contents can be described by three major phases, push phase, stop-and-copy phase, and pull phase [6]. In the literature, two techniques are identified for memory state migration, the post-copy and pre-copy methods, with the most preferable being the pre-copy technique. The pre-copy method is a combination of the push and stop-and-copy phases that most hypervisors (KVM, VMware and XEN) utilize and prefer for a Live migration execution. As illustrated in Figure 8, first all the memory pages are transferred in an iterative fashion whilst the source host is still running. The process continues until only a small amount of memory pages are left where the stop-and-copy phase takes over; the source machine is then halted while the dirty pages and CPU state are transferred to the destination host. After that, operation continues at the destination machine [6].

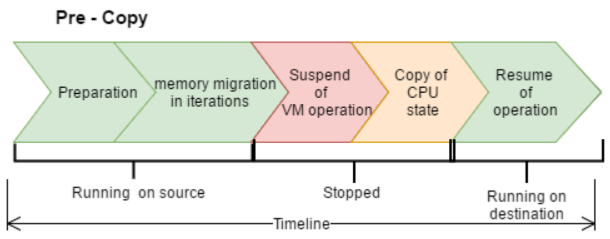

Figure 8: Memory Live migration techniques

### 3) *I/O device state and Interrupt Controller*

The most challenging aspect of the BMLM process is migration of the I/O and network device states. In general, a device changes its state either by the OS changing the operational configuration or by the devices themselves, like a rebooting process, where a device changes status until it is ready for use [13]. However, some of the device registers are not readable or writable so a hypervisor, even if it can read those registers, may not be able to write or modify their content at the destination machine. Likewise, even if the hypervisor can write a value into a destination device register it may not be able to read the content on the source in the first instance.

The ARM architecture introduces the Generic Interrupt Controller architecture for handling all interrupt sources for any processor connected to a GIC. GICv2 and later, includes the GIC Virtualization extensions where a hypervisor can either handle a physical interrupt itself or a virtual interrupt that is signaled to a virtual machine. A GIC is composed by two major components, the distributor and one or more CPU interfaces as the number of the processors. The distributor performs interrupt prioritization and distribution of the interrupt to the corresponding CPU interface whilst a CPU interface performs priority masking and preemption handling for a connected processor on the system [12, 17].

In contrast to x86 architectures, ARM makes use of only the MMIO (Memory-mapped I/O) method to perform and handle input/output (I/O) requests between the CPU and peripheral devices [17]. So, similar to the BitVisor approach, in our implementation we will try to obtain any unreadable device states by configuring the hypervisor to monitor communications between the OS and peripheral devices. When the OS makes a request to access or write into an I/O address, the hypervisor will intercept this communication to obtain the value of those registers. Moreover, to set the unwritable device registers, we have to configure the hypervisor at the destination machine to collaborate with the source machine in a way so we can manipulate and control physical devices to change their state to the desired settings.

## VI.Conclusion

Edge computing solutions introduce a remarkable shift, bringing processing power closer to the edge users and increasing the performance of storage and computation of latency-sensitive data, instead of consolidate it them into a centralised data centre on Cloud. Micro-datacentres (MDCs) are resources-rich rack-size systems located at the edge of a network, processing the data that generated by a variety of IoT devices. However, those micro-datacentres leverage server virtualisation characteristics and benefits which creates significant performance degradation making them unsuitable platforms for "Big-data"-type applications. Therefore, the emergence of bare-metal edge servers, free of virtualisation, offering native high performance processing dedicated to specific tasks become an attractive alternative implementation for micro-datacentre infrastructures. We introduced a proposal for the development of a practical approach for live migration performed on bare-metal micro-datacentres by leveraging the capabilities of new ARM CPU architecture. Our goal is the utilization of a very thin hypervisor to perform live migration of the physical states without the need for a heavyweight virtualization, eliminating the virtualization overhead.

## Acknowledgment

The work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/P004407/1 and EP/P004024/1.

## References

[1] M.a. Kozuch, Michael Kaminsky, and M.P. Ryan. "Migration without virtualization". In: *Proceedings of the 12th conference on Hot topics in operating systems* (2009), pp. 10–10.

[2] Takahiro Shinagawa et al. "BitVisor". In: *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments - VEE '09* (2009), p. 121.

[3] Jui Hao Chiang, Maohua Lu, and Tzi Cker Chiueh. "Physical machine state migration". In: *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS* (2011), pp. 25–32.

[4] Jui-Hao Chiang, Maohua Lu, and Tzi-cker Chiueh. "Bootstrapped Migration for Linux OS". In: *Proceedings of the 8th ACM International Conference on Autonomic Computing* (2011), pp. 209–212.

[5] Prashant Varanasi and Gernot Heiser. "Hardware-Supported Virtualization on ARM". In: *ApSys'11* (2011), p. 11.

[6] P. Getzi Jeba Leelipushpam and J. Sharmila. "Live VM migration techniques in cloud environment - A survey". In: *2013 IEEE Conference on Information and Communication Technologies, ICT 2013* Ict (2013), pp. 408–413.

[7] Bhanu Prakash Reddy Tholeti. *Hypervisors, Virtualization, and Networking*. Elsevier Inc., 2013, pp. 387–416.

[8] Paul Burns. "The IT benefits of bare metal clouds". In: (2014).

[9] David S Linthicum and Virginia Senf. *Leveraging bare metal clouds*. Tech. rep. 2014, p. 13.

[10] Kim. "Prototype of Light-weight Hypervisor for ARM Server Virtualization Young-Woo Jung, Song-Woo Sok, Gains Zulfa Santoso, Jung-Sub Shin, and Hag-Young". In: *Embedded Systems and Applications 215* (2015).

[11] Yushi Omote, Takahiro Shinagawa, and Kazuhiko Kato. "Improving Agility and Elasticity in Bare-metal Clouds". In: *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '15* 50.4 (2015), pp. 145–159.

[12] Christoffer Dall et al. "ARM Virtualization: Performance and Architectural Implications". In: *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016* (2016), pp. 304–316.

[13] Takaaki Fukai et al. "OS-Independent Live Migration Scheme for Bare-Metal Clouds". In: *Proceedings - 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, UCC 2015* (2016), pp. 80–89.

[14] Weisong Shi et al. "Edge Computing: Vision and Challenges". In: *IEEE INTERNET OF THINGS JOURNAL* 3.5 (2016). URL: http://www.cs.wayne.edu/%7B~%7Dweisong/papers/shi16-edge-computing.pdf.

[15] Jeffrey Burt. *ARM Server Chips Challenge X86 in the Cloud*. 2017. URL: https://www.nextplatform.com/2017/02/01/arm-server-chips-challenge-x86-cloud/ (visited on 08/17/2017).

[16] Basic Architecture and Order. *Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3C: System Programming Guide, Part 3*. 2536.

[17] *ARM Information Center*. URL: http://infocenter.arm.com/help/index.jsp (visited on 08/13/2017).

[18] "Micro Data Center Solutions Micro Data Center Environments". In: (). URL: https://www.anixter.com/content/dam/anixter/resources/brochures/anixter-micro-data-center-brochure-en.pdf.