**Prof. Alain Tchana - ERODS Reseach team - LIG Laboratory - Grenoble - France**
https://perso.ens-lyon.fr/alain.tchana/
2022
alain.tchana@grenoble-inp.fr

# TGVM: Live migration of virtual machines on heterogeneous processor architectures of the same ISA.

**Key words :** Virtualization, Computer Architecture, Emulation, Binary Instrumentation.

## Datacenter organization and virtual machines (VM) live migration: rack homogeneity and homogeneous migration?

A data center is mainly composed of machines equipped with processors having the same instruction set architecture (ISA) (x86 or ARM). Currently, Intel dominates the world of data centers, even if ARM ARM is starting to interest data center operators because it offers better energy consumption. Therefore, a typical data center will mainly consist of Intel machines and some ARM machines. Several generations of processors with the same ISA can be present in the same data center for scalability reasons (to meet the needs of the needs). Finally, the machines in the data center are organized in racks (cabinets of a dozen machines). Each rack comprises homogeneous machines: processors of the same ISA and the same generation. To learn more about the organization of a data center, see the paper (figure 1) published at OSDI in 2020 "Protean: VM Allocation Service at Scale"[7]. Due to the architecture mismatch of processors of different generations, live migration is currently impossible. Live migration is only performed between homogeneous machines (within the same rack). The only existing approach to enable migration between different architectures consists of downgrading the functionalities of the most recent processor to the functionalities of the oldest processor. This is what Intel proposes with FlexMigration. From our discussions with Outscale (a cloud operator), this solution is never used because it ignores the reason for acquiring new processors. Indeed, FlexMigration should be activated at the start of the rack cluster.

## Scientific challenges: how to hot-match two different processor architectures?

The inability to migrate a VM from architecture A to a different architecture B is not a problem of incompatibility of hypervisors concerning architectures. The hypervisors can run the VM on both architectures A and B. The problem is the live migration of the VM state from A to B. Indeed, this state includes the state of the processors. To better illustrate the problem, if A contains 20 registers and B contains only 15 registers, the problem is how to map the 20 registers of A to the 15 of B? With Intel's FlexMigration, when machine A starts up (before the VMs start-up), the BIOS configures it to use only 15 registers like B. Thus, the future migration of a VM from A to B will be possible. The manufacturers feel that they have brought a solution to this problem. Nevertheless, this solution is not practical. Hypervisor developers feel that this problem must be solved at the architecture level. With TGVM, we are exploring two approaches: software (so it can be used quickly) and hardware (it can be used in the long term).

## Current status of the project

Our initial studies reveal several findings that support our ambition. First, when the subset of processor features that an application running in a VM uses is the same between two physical machines with different processors, it is possible to "force" the live migration of the VM. Indeed, we realized that current hypervisors and cloud providers present at VM startup all the processor's features to the VM, even if the latter does not use all these features. However, VM's operating system ingests these features at VM startup. In some cases, this unnecessary strong link prevents live migration of the VM in the context of processor heterogeneity. The first line of research for TGVM (which we call processor functionality minimization) will be to provide an automatic static discovery tool (before a VM and its hosted application services are put into production) of all the processor functionalities it needs. Thus, the VM and its operating system will be presented only by the hypervisor with the subset of processor functionalities that the applications will need. In this way, we increase the proportion of heterogeneous machines that can host the VM during a hot migration. The challenge with providing this tool for discovering the functionality needed by the applications in a VM is the black-box nature of VMs. How do we inspect this black-box to exhaustively (with as few false positives and negatives as possible) discover the necessary processor functionality?

The second line of research (which we call emulation) addresses the problem of hot migration of the VM when its subset of processor functionality is not fully present on the destination machine. The basic idea is that some processor functionality can be provided in software ways either at the hypervisor level, the VM's operating system, or the applications that the VM runs. Indeed, for each processor functionality, we assume it

is possible to find or implement a software alternative. This assumption is viable since introducing processor features is always done as a replacement for a software version. This observation is remarkably accurate in virtualization, where hardware-assisted virtualization lags behind software innovations. For example, Intel PML (Page Modification Login), which allows tracking of memory pages in the processor, replaces its software alternative based on write protection of pages and page fault triggering. We can also mention Intel CAT (Cache Allocation Technology), which replaces the page coloring used in software to partition the processor caches. This emulation approach can also be made using FPGAs, which are hot-swappable boards. Thus, a processor functionality not present on the destination machine can be hot-programmed in the FPGA (associated with each physical machine) before the VM arrives. Note that these two types of emulation, although allowing the hot migration of the VM between heterogeneous architectures, will introduce a performance degradation that we hope will be acceptable.

## Intership work

The recruited student will explore the first research axis we mentioned above, i.e., to implement a tool for automatically discovering the set of CPU features required for executing a given application in a VM. For this, we can explore two approaches. A disable-and-try approach which consists in disabling a feature and see if the application in the VM runs without any problem. If it does, then the disabled feature is not needed. This approach, therefore, requires exploring the feature space and their combination, which might not be realistic (to be seen). The second approach consists of analyzing the applications' binaries to discover the features used. For this, tools such as angr will be considered. The work on discovering syscall in a binaries could also inspire us.

## References

[1] https://tel.archives-ouvertes.fr/tel-00068012/document
[3] https://fr.outscale.com/
[4] https://unikraft.org/
[5] https://unikraft.org/community/hackathons/2022-05-lyon/
[6] https://2021.eurosys.org/papers.html#papers
[7] https://www.microsoft.com/en-us/research/uploads/prod/2020/10/ProteanCR.pdf