FPGA based system for the acceleration of Cloud Microservices

Julien LALLET and Andrea ENRICI and Anfel SAFFAR
Software Defined Mobile Network
Nokia Bell-Labs France
Nozay, France
firstname.lastname@nokia-bell-labs.com

***Abstract*—Scalability, distributivity, interoperability, modularity introduced in cloud computing have deeply changed the legacy data center architecture, implementation and processing capabilities. The atomic network services offered by cloud architectures are called microservices. Unlike virtual machines, microservices can be implemented in the form of low resources footprint applications as containers (Docker, LXC etc.) or even smaller as unikernels (IncludeOS, ClickOS, Rumprun, HermitOS etc.). The need to efficiently offload the processing of computation-intensive applications has motivated the introduction of Field Programmable Gate Arrays (FPGA) boards in servers. FPGAs can nowadays be considered as cloud standard processing resources.**

However, in today cloud data centers, FPGAs cannot be accessed to run concurrent microservices. This severely limits the efficient deployment of microservices. This paper aims at introducing an FPGA-based system for the concurrent acceleration of cloud-native microservices onto FPGAs. We also provide a brief state of the art on cloud hardware processing based on FPGAs and their limitations.

***Index Terms*—FPGA, cloud, microservice, acceleration**

I. Introduction

In the last decade, successive transformations in the infrastructure and architecture of legacy data centers have lead to a new type of cloud systems. Scalability, distributivity, interoperability, modularity are some of the main characteristics which make today's data centers being cloud systems. To reach this target, it has been necessary to invent and develop new tools with new features as for example management and orchestration softwares. ETSI has standardized the Cloud framework as ETSI NFV [1]. The current NFV framework is based on three main elements: Network Functions Virtualization Infrastructure (NFVI), VNF (Virtual Network Functions) and MANO (MANagement and Orchestration). VNFs are implemented on top of NFVI as Xen [2] or KVM [3], in the form of virtual machines, managed and orchestrated by dedicated tools. The community is working on enhancing the framework to support more efficient virtualization like containers (as Docker [4]). The container management ecosystem is now very rich. One can cite, Swarm, Mesos [6], and Kubernetes [5] which is gaining more and more attraction and could become the de-facto standard in the orchestration. Since server processors are essentially based on Generic Purpose Processors (GPPs) as x86 or ARM architecture, the need for efficient processing was first filled by the use of Graphic Processing units (GPU). Originally designed for video processing offloading, GPU architectures are perfect for image and video processing, or more generally for matrix operations. However, if other types of data processing are needed (signal processing, ciphering etc.), performance are not improved by the use of GPU, even most probably decreased due to the specific architecture of GPUs [7]. To achieve the required processing efficiency, cloud resource providers as Amazon [8] or Microsoft [9] and new cloud actors as Accelize [10] have added Field Programmable Gate Arrays (FPGA) in their data centers. FPGAs can be configured to process specific applications, as this is the case for Application Specific Integrated Circuits (ASIC), except that FPGAs can be reprogrammed at runtime to modify the functionality of the hardware circuitry. Using FPGAs in cloud infrastructures allows to fill the gap between the processing flexibility offered by GPPs and the efficiency which can be achieved by ASIC. The well spread use of FPGAs in data centers nowadays allows us to consider these processing resources as standard solutions for acceleration.

However, the way FPGAs are used today does not permit to consider them as components that are fully integrated in the microservices and Cloud paradigms. The microservice approach allows to simplify complicated software systems by breaking them into sub-components and distributing these components across many computing servers. In this approach, an application consists of many small independent services, each service is running on its own independent process. The introduction of microservices in cloud infrastructure supports modularity, flexibility and distributed software components. Today's FPGAs available in a cloud infrastructure [8]–[10] are not concurrently shared among running microservices on the same host as it is the case for memory or Central Processing Unit (CPU) resources. In this extended abstract, we will present our current work on the design of a shared FPGA based system for the acceleration of concurrent microservices.

II. FPGA based system for the acceleration of Cloud Microservices

In this paper, we introduce a novel approach to break the silos between the accelerated infrastructure and the software

Acknowledgement: This work has been carried out with the support of the EU project NGPaaS funded from the European Unions Horizon H2020-ICT-2016-2017 Programme under grant agreement n 761557

that should be executed following microservice principles. The specificity of this accelerated node is its capability to run microservices that are able to access and share connected FPGAs for function offloading and acceleration. The FPGA based infrastructure is detailed in Fig. 1 and composed of two resources. First, a cloud server is used for the processing of virtual functions in the form of containers. The second resource is a FPGA board connected to the server through a PCIe interface. To provide flexible and shared acceleration resources, the FPGA is split into several acceleration slots which are directly and independently accessible from containers through the PCIe interface for data processing. In order to grant maximum throughput to the data plane, the control plane passes through IP network.

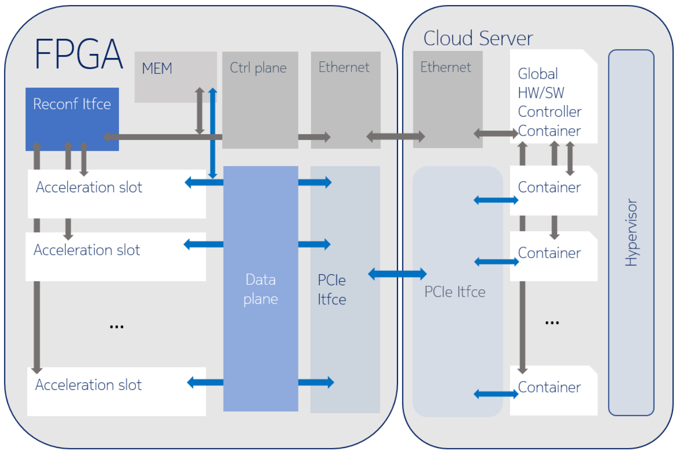

Fig. 1. Overview of the architecture

In the following, we detail the four steps needed by a container to access a FPGA on the infrastructure depicted Fig 1. We decided to illustrate the example with a docker running on a KVM server which would have to offload data to the FPGA for efficient data processing.

a) Docker instantiation: The instantiation of an accelerated docker is done through the standard docker process. There is no need for specific requirements or docker add-ons. Access to the PCIe interface is granted at this step by the docker launch command.

b) FPGA partial reconfiguration: The configuration of each acceleration slot is done by partial reconfiguration which is internally controlled by the FPGA. To enable the partial reconfiguration from the docker instantiation, the docker image has been configured to launch a reconfiguration TCP client application which will send the bitstream to the TCP server running on a Linux OS, embedded in the FPGA. The FPGA configuration controller proceeds to the partial reconfiguration internally through the Xilinx PCAP interface.

c) Docker running: Once the FPGA acceleration slot has been configured for a given Docker, the latter can send data packets to the FPGA for the accelerated data processing. At the same time, the use of the other acceleration slots is possible. Each such a slot can be managed and used independently from any other docker or any other kind of microservice running on the KVM server. We could successfully run four independent acceleration slots from four different dockers without any data throughput degradation. Each docker can send and receive data to the FPGA at a maximum throughput of 1 GBytes/s for processing that is accelerated inside the FPGA. On the control plan, the reconfiguration throughput on the PCAP interface is around 124 MB/s giving a reconfiguration time of 24,2 ms for the partial bitstream used for one acceleration slot.

III. Conclusion

In this extended abstract, we have presented the design of a FPGA based system on which we can run concurrent microservices with access to a shared FPGA as a local accelerator. We have successfully run acceleration resources inside the FPGA shared between any multiple microservices running on the connected server. The functionality implemented in the FPGA hardware can be changed at runtime thanks to the FPGA's partial reconfiguration capabilities. In the future, first, we plan to provide more flexibility to access any FPGAs inside a data center from any running server. In parallel, we also plan to create FPGA API for orchestrators to better accelerate scheduling and management functionalities in virtualization frameworks.

References

[1] ETSI GS NFV 002, "Network functions virtualization (NFV); architectural framework v1.1.1," ETSI, Tech. Rep., October 2013.
[2] D. E. Williams, *Virtualization with Xen(Tm): Including XenEnterprise, XenServer, and XenExpress: Including XenEnterprise, XenServer, and XenExpress*. Syngress Publishing, 2007.
[3] I. Habib, "Virtualization with kvm," *Linux J.*, vol. 2008, no. 166, Feb. 2008.
[4] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014.
[5] D. K. Rensin, *Kubernetes - Scheduling the Future at Cloud Scale*, 1005 Gravenstein Highway North Sebastopol, CA 95472, 2015. [Online]. Available: http://www.oreilly.com/webops-perf/free/kubernetes.csp
[6] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 295–308. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972488
[7] S. Kestur, J. Davis, and O. Williams, "Blas comparison on fpga, cpu and gpu," in *IEEE Computer Society Symposium on VLSI*. IEEE, July 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/blas-comparison-on-fpga-cpu-and-gpu/
[8] Amazon. (2017) Run customizable fpgas in the aws cloud. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/
[9] A. M. Caulfield *et al.*, "Configurable clouds," *IEEE Micro*, vol. 37, no. 3, pp. 52–61, 2017.
[10] Accelize. (2017) Enabling fpga-acceleration-as-a-service. [Online]. Available: https://www.accelize.com