When FPGA Meets Cloud: A First Look at Performance

Xiuxiu Wang, Yipei Niu, Fangming Liu*, *Senior Member, IEEE,* and Zichen Xu, *Member, IEEE*

Abstract—Cloud service providers promote their new field programmable gate array (FPGA) infrastructure as a service (IaaS) as the new era of cloud product. This FPGA IaaS wraps virtualized compute resources with FPGA boards, e.g., Amazon AWS F1, and reserves acceleration capability for specific applications. Though this acceleration technique sounds promising, questions like real world performance, best-fit scenarios, portability, etc., still need further clarification. In this paper, we present one of the first few empirical studies that take a close look at FPGA clouds from the tenants' perspective. We have conducted measurement studies on Amazon AWS, Alibaba, and Huawei clouds for over one year. The experimental results show that: (1) Tenants experience severe performance-cost imbalance on FPGA IaaS platforms; (2) The inter-communication performance in FPGA clouds is tightly constrained by hardware drivers, e.g., small optimization of DMA drivers for PCIe can harvest significant performance gain; (3) The virtualized FPGA clouds are far from mature, e.g., small-sized jobs can greatly degrade the performance of FPGA clouds due to underutilized PCIe bandwidth. Our study not only provides useful hints to help tenants with FPGA service selection, but also sheds some lights for cloud providers to improve the performance of FPGA clouds.

Index Terms—FPGA Cloud, FPGA Acceleration, Virtualization, Performance Measurement.

1 Introduction

Field Programmable Gate Array (FPGA), is a re-programmable silicon chip that enjoys both high throughput and low latency by breaking the paradigm of sequential execution and accomplishing more per-clock. Benefiting from this powerful computing capacity and "hard" implementation of program execution, FPGA is widely used to accelerate applications like image recognition [35], spam filtering [14], complex data analysis [26], [27], etc.

Besieds FPGAs, GPUs are also applied for high performance acceleration. However, compared to GPU, FPGA can provide superior energy efficiency, which is an essential metric especially in large-scale datacenters. In [34], with matrix multiplication workloads, the authors point out that Stratix 10 FPGA achieves nearly 90% performance and 200% performance/watt of TitanX Pascal GPU. Similarly, it is shown that EVGA GeForce GTX 295 GPU consumes one order of magnitude higher energy than Stratix III FPGA with digital signal processing applications, and its performance is averagely three times higher [22]. Increasingly, cloud service providers (CSP) prefer this new managed FPGA infrastructure that masks the difficulty of FPGA programming, yet effectively adapts its computation feasibility. Intel, with its $16.7 billion acquisition of Altera, predicts that 30% of servers will have FPGAs by 2020 [7]. AWS F1 [3], Alibaba Cloud FPGA instances [1], Huawei FACS [8], and Tencent FPGA Cloud Computing [2], are examples of commercial FPGA IaaS.

FPGA IaaS can lease bundles of memory, CPU, and a specific FPGA board, called FPGA instance, from IaaS clouds. For example, Amazon designed a cloud server with a fabric of pooled accelerators that interconnects with up to 8 FPGAs [3]. Each FPGA is connected with the cloud server via the PCIe network. The interconnection of FPGAs allows the chips to share memory, and achieves high bandwidth as well as low latency for inter-chip communication, thus supporting up to 8 FPGA boards.

However, from a tenant's perspective, adapting FPGA instances could be expensive for two reasons. First, an FPGA instance can virtualize compute resources but cannot virtualize inter-chip communication. An ill-managed PCIe bus can significantly diminish the performance gained from the FPGA accelerator. Furthermore, public cloud is a multi-tenant system with contention and interference, which may lead to severe performance variation. Second, concerning virtualization and multi-tenancy of public cloud, tenants are required to develop a different strategy of using FPGA IaaS compared to dedicated physical FPGA, so as to achieve high performance and low cost. Consequently, cloud tenants may have the following questions: 1) What are advantages and disadvantages of FPGA IaaS; 2) Based on sufficient knowledge of FPGA IaaS, how to cost-effectively rent cloud FPGA to maximize profit, i.e., achieving high acceleration

- *This work was supported in part by NSFC under Grant 61722206 and 61761136014 (and 392046569 of NSFC-DFG) and 61520106005, in part by the National Key Research & Development (R&D) Plan under Grant 2017YFB1001703, in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFKJXX009 and 3004210116, in part by the National Program for Support of Top-notch Young Professionals in National Program for Special Support of Eminent Professionals, in part by the National Science Foundation China (NSFC) under Grant 61702250.*
- *X. Wang, Y. Niu, and F. Liu are with the National Engineering Research Center for Big Data Technology and System, the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan, 430074, China. E-mail: {m201672794, niuypei, fmliu}@hust.edu.cn. The corresponding author is Fangming Liu.*
- *Z. Xu is with the Generic Operational and Optimal Data Lab, Nanchang University, 999 Xuefu BLVD, IEB A608-1, Nanchang, 330000, China. E-mail: xuz@ncu.edu.cn.*

performance with minimal cost.

CSPs do expose APIs or even provide manufactured tools to evaluate their cloud products. For example, Amazon CloudWatch helps to monitor system-wide performance, and Alibaba Pricing Calculator helps to estimate the cost of renting the instances. However, these mature tools are built for virtualized resources, like memory or CPU, rather than FPGA. Related metrics of FPGA, such as block counters and steamed bytes, are hidden in the higher level abstraction. We need to understand the real performance of FPGA cloud from the tenants' perspective, so we come up with a fine-grained approach to achieve this.

It is challenging to conduct performance analysis of cloud FPGA instances from the tenants' perspective. First, although the abstraction of FPGA programming is easy for development, it is difficult to understand the performance fundamentally. Second, CSPs conceal the details about the architecture design of FPGA IaaS, making it hard to understand the data movement along memory hierarchies. Third, the complicated nature of virtualized FPGAs offered by multiple vendors increases the difficulty in both program migrations and program analysis. To address the challenges, we conduct an empirical measurement on FPGA clouds, motivated by a measurement study about the performance characteristics of FPGA IaaS for various compute-intensive and streaming workloads. In an effort to help tenants understand the real performance of cloud FPGA instances, our measurements can be divided into three levels specifically:

- **VM level:** We explain the rationales behind the abstraction of programming interfaces related to operating FPGA, i.e., device drivers. Our discussion about the device drivers will aid tenants in understanding how they work and improving FPGA acceleration performance. The performance metrics of device drivers are defined as PCIe throughput and latency of data transmission.
- **Inter-communication level:** We trace the data movement along memory hierarchies in FPGA clouds and measure the PCIe bandwidth variation, so as to identify and quantify the data transfer overhead introduced by virtualization techniques for tenants.
- **Device level:** We seek to explore how the cloud-based FPGA board performs when accelerating various applications, so that tenants can select proper FPGA instances based on their application types. The performance metrics of FPGA board are defined as throughput and latency of acceleration jobs.

In this paper, we take a first effort to empirically quantify the FPGA instance performance of three major cloud providers, namely, Amazon AWS, Alibaba and Huawei clouds over one year. Our findings can serve as a basis to understand performance of cloud FPGA instances. We also expose the opportunity for tenants towards getting the maximum benefit. Here we list our findings:

1) **There exists severe performance-cost imbalance on the FPGA IaaS platforms.** Overall, the low-end FPGA instances produce as higher as 1.4× performance-cost ratio (i.e., performance gain against CPU divided by cost during acceleration) compared to the high-end ones, just bounded by 20% performance degradation on average.
2) **Streaming workloads fail to fully exploit the advantages of FPGAs and deliver extremely low performance.** Compared to compute-intensive workloads, we discover that streaming workloads perform 20× to 40× slower due to the inefficiency of data communication between FPGA and VM in FPGA clouds.
3) **The number of FPGA nodes (or compute units) and data communication bandwidth differ in ways that can significantly affect performance.** In a distributed FPGA acceleration system, only striking a balance between the data processing speed and the data transfer speed can achieve the best performance of FPGA clouds.
4) **Drivers of FPGA devices show a large variance in performance.** Specifically, among the cloud FPGA instances we evaluate, for communications between FPGA and VM, user space driver outperforms kernel space driver, increasing throughput by 1.6× at most when transfer data size exceeds 4M during writing to FPGA device. As for AWS FPGA instances, data transfer performance improves as the device driver evolves.
5) **A significant overhead of communications between FPGA and VM is introduced in FPGA clouds.** The interconnection bandwidth in FPGA IaaS cloud is lower than that in a physical environment due to extra memory copy and immature virtualization, resulting in at least a 10% decrease.
6) **Acceleration performance is mostly determined by the capacity of a certain type of resource on an FPGA board in FPGA IaaS clouds.** For instance, we observe that for floating-point arithmetic intensive applications, such as deep learning, digital signal processor is crucial to the overall acceleration performance.

The contributions of this paper are as follows. First, we provide a fine-grained approach, including metrics, benchmarks, and tool kits, to evaluate FPGA IaaS from VM, inter-communication, and device levels. Second, based on the approach, we conduct a series of measurements to demonstrate the real performance of multiple major FPGA IaaS platforms. Third, according to the findings, we produce multiple guidelines and takeaways for FPGA IaaS tenants.

The rest of this paper is organized as follows. Section 2 introduces related work. In Section 3, we characterize the performance of FPGA IaaS and introduce its infrastructure. Section 4 shows the methodology of characterizing performance of FPGA clouds. Section 5 demonstrates how device drivers impact on data transfer efficiency between VM and FPGA. Section 6 presents the PCIe virtualization overhead in FPGA IaaS. In Section 7, we show the results of accelerating three benchmarks demanding for different resources on an FPGA. Section 8 concludes the whole paper.

2 Related Work

FPGA in the Datacenter. Putnam *et al.* first study the FPGA deployment in real large-scale datacenters, i.e., the Catapult project [38]. They distribute FPGAs across servers and present a high throughput and low latency ranking workload with resilience. Inspired by this work, several literature studies how to apply FPGAs at warehouse scale for big-data processing, network enhancing applications. Weerasinghe *et al.* present a network-attached FPGA integration in datacenter infrastructure [41]. Lockwood *et al.* implement

a memcached system with ultra low latency by offloading the KVS to FPGA in [30]. Huang *et al.* abstract FPGA accelerators as a service and provides programming and runtime environments for large-scale FPGA deployment in datacenters [24]. An FPGA/Spark system is implemented in [20] to accelerate DNA sequencing. Li *et al.* propose a CPU-FPGA co-design framework to accelerate network functions in [29].

FPGA in IaaS. Recent works have made the effort to build cloud integrated with FPGAs. Chen *et al.* design a prototype that virtualizes FPGA and make it available for multiple tenants in [19]. Utilizing the OpenStack framework, Byma *et al.* abstract FPGA in a resource pool, where tenants can rent them as general cloud computing units [17]. Asiatici *et al.* design a runtime system that enables FPGAs integrated in the cloud with marginal performance overheads [15].

FPGA in SaaS. Based on [38], Microsoft Catapult project builds a new cloud architecture that uses FPGAs to accelerate Bing web search services as well as networking [18]. This design improves the datacenter deployment and can scale up two order of magnitude nodes in contrast to the first infrastructure. Recently, Microsoft company presents AccelNet [21], an FPGA-based platform for host SDN processing, supported by the software and hardware infrastructure of the previous Catapult work. Azure's new cloud offering for FPGA-based machine learning platform enables customers utilizing drag-and-drop controls to deploy machine learning services [5].

3 Characterizing FPGA IaaS Performance

Cloud FPGA instances are leveraged to accelerate various applications for FPGAs' hardware-level parallelism. In this section, first, we reveal performance variation and performance-cost imbalance in FPGA clouds. Second, we stress the importance of striking a balance between compute capacity and data communication in FPGA clouds. Based on the above investigations, we further dive into FPGA clouds to explore which factors and how they impact performance.

3.1 Single-node Acceleration Performance of FPGA IaaS

CSPs promise their cloud FPGA instances can achieve high performance with FPGA acceleration. For example, Amazon claims AWS FPGA instances can provide up to 100× acceleration compared to CPUs for compute-bound applications, such as genomic analysis [3].

Longing for the high performance offered by FPGA accelerators, cloud tenants expect to outsource certain applications to FPGA IaaS. By offloading heavy computing jobs to FPGA, cloud tenants can accelerate their applications. However, the true acceleration performance of FPGA instances still remains unknown. Furthermore, little is known about the acceleration performance of other types of applications, such as streaming jobs. As a result, to reveal the real acceleration performance for different types of applications on FPGA IaaS platforms, we select two compute-intensive applications (i.e., digital recognition and optical flow) and two streaming applications which are communication-intensive (i.e., spam filtering and face detection) to evaluate the acceleration performance of various FPGA instances. These applications are typical benchmarks for FPGA acceleration [44]. We conduct the measurements on four FPGA instances: Amazon EC2 F1.2xlarge (AWS F1), Alibaba Cloud ECS.f2-c8f1.2xlarge (Ali F2), Alibaba Cloud ECS.f3-c8f1.2xlarge (Ali F3) and Huawei Cloud fp1c.2xlarge.11 (HW F1). Detailed specifications can be found in Table 1. All of our experiments are conducted in cloud FPGA environments, except a few baselines against FPGA IaaS.

Dataset. The digital recognition application trains digit classification models with K-nearest-neighbor (KNN) algorithm. 20,000 samples of digits are selected from MNIST [12] dataset (18,000 samples for training and 2,000 samples for testing). The spam filtering application uses logistic regression models for classification. In the measurement, we select 5,000 emails from Apache SpamAssassin Dataset [4], where 4,500 emails are used for training and 500 emails are used for validation. The optical flow benchmark utilizes Lucas-Kanade method [42] to detect objects in sequential image frames. The dataset is MPI Sintel [16]. ViolaJones algorithm [40] is employed to complete the face detection. All the algorithms of benchmarks are hard-coded in cloud FPGA for acceleration in advance.

Method of Measuring Latency. We prepare datasets and write them to an FPGA board for acceleration in each round of experiment. Then the FPGA accelerator is launched by invoking `runKernels` of OpenCL library. Before transferring the data to the FPGA and after retrieving the results from the FPGA, we use `clock_gettime` system call to get start and end time respectively, so as to calculate execution time on the FPGA (including computation and data transfer latency). Throughput is derived by dividing the amount of data with the execution time. Each experiment is repeated 10 times every 24 hours, lasting for 14 days.

Performance Speedup. To evaluate the acceleration performance of cloud FPGA instances, we define performance speedup as f/c, where f is the throughput of cloud FPGA solutions, and c is the throughput of CPU solutions. For the CPU solutions, we run the applications on the compute instance of AWS F1 without offloading jobs to FPGA board.

Performance-Cost Ratio. From the perspective of a cloud tenant, cost efficiency is another major concern when selecting cloud FPGA platforms. Hence, to evaluate the cost-efficiency, we define a metric of performance-cost ratio as $f/(p \times t)$, where p is the hourly price of AWS F1 and t is the execution time.

Figure 1 and Figure 2 plot the comparison of performance speedup and performance-cost ratio on each platform (performance-cost ratio is normalized to Ali F3).

Performance Variations in FPGA IaaS. As shown in Figure 1, on average, the two compute-intensive workloads, i.e., digital recognition and optical flow, see 40× and 20× performance speedup, respectively. On the contrary, the two communication-intensive workloads, i.e., spam filtering and face detection, only deliver no more than 2× performance speedup, which is far less than those of the compute-intensive applications. It is because that the performance speedup of different applications depends on their parallelism and memory footprint. For example, digital recognition achieves the highest performance speedup since it

TABLE 1: VM, PCIe and FPGA specifications of each cloud FPGA instance.

Instance Type	VM			PCIe	FPGA	Price
	Physical Processor	vCPU	Hypervisor	Specification	FPGA Card	Hourly Price
Amazon EC2 F1.2xlarge (AWS F1)	Intel Xeon E5-2686v4	8	Xen	PCIe 3.0x16	Xilinx VU9P	$1.65 per hour
Alibaba Cloud ECS.f1-c8f1.2xlarge (Ali F1)	Intel Xeon E5-2682v4	8	KVM	PCIe 3.0x8	Intel Arria 10	$0.87 per hour
Alibaba Cloud ECS.f2-c8f1.2xlarge (Ali F2)	Intel Xeon E5-2682v4	8	KVM	PCIe 3.0x8	Xilinx KU115	$1.05 per hour
Alibaba Cloud ECS.f3-c8f1.2xlarge (Ali F3)	Intel Xeon Platinum 8163	8	KVM	PCIe 3.0x16	Xilinx VU9P	$2.06 per hour
Huawei Cloud fp1c.2xlarge.11 (HW F1)	Intel Xeon E5-2697v4	8	KVM	PCIe 3.0x16	Xilinx VU9P	$1.49 per hour

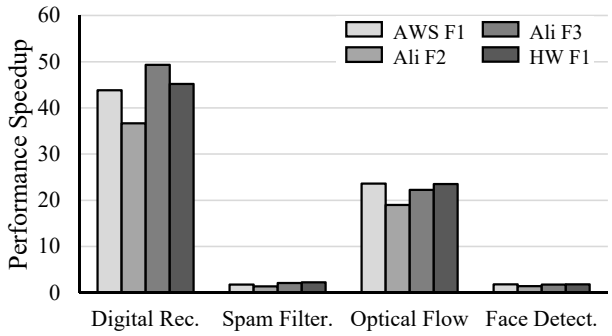

Fig. 1: Performance speedup of each cloud FPGA instance.

has high data-level and instruction-level parallelism by unrolling and pipelining the loop. Whereas, spam filtering transfers large volumes of data from host to FPGA resulting in a high communication to compute ratio, i.e., the data transfer time is much longer than data processing time in FPGA. So the data communication bandwidth between FPGA accelerator and host becomes the performance bottleneck. Noting that although AWS F1, Ali F3, and HW F1 have the same FPGA card and similar CPU configuration, their performance varies significantly. Such a performance gap across CSPs may result from their respective virtualization solutions. In addition, Ali F2 has the lowest hardware configuration and performs the worst, with a 23% performance degradation on average.

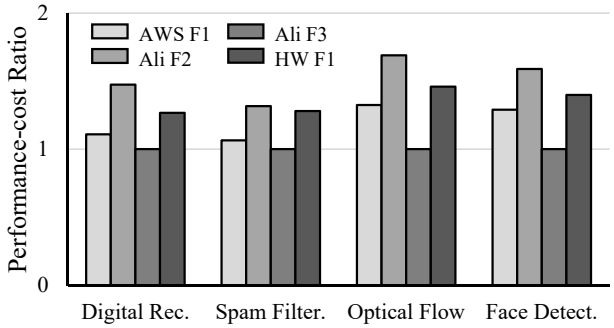

Fig. 2: Performance-cost ratio of each cloud FPGA instance.

Performance-cost Imbalance in FPGA IaaS. Figure 2 illustrates that, overall, Ali F2 (lower-end FPGA instance) produces higher performance-cost ratio (around 1.4×) compared to AWS F1, Ali F3, and HW F1 (higher-end FPGA instances), just bounded by 20% performance degradation on average. HW F1 has the highest performance-cost ratio compared with AWS F1 and Ali F3 that have similar hardware configuration and comparable performance.

Discussion. These results indicate a market issue of FPGA IaaS, i.e., there exists severe performance-cost imbalance in FPGA IaaS cloud. For specific applications, the acceleration performance shows variation with the same cost. To adapt such performance-cost imbalance, cloud tenants are required to develop proper strategies when renting FPGA instances. We suggest that, for applications with strict requirements for latency or throughput, such as real-time video processing, tenants are supposed to rent the high-end cloud FPGA instances to meet their requirements. On the contrary, tenants may rent the low-end ones to save cost.

3.2 Multi-node Acceleration Performance of FPGA IaaS

Distributed computing in public cloud has attracted attention in both industry and academia. When a workload outreaches the compute capacity of the hardware, using scale-out solutions to meet demand is a key issue for tenants. There have been many studies concentrating on exploring the performance of infrastructure scalability for rapidly growing data volumes [13], [31]. The performance of scale-out systems with accelerators is a fundamental problem to be revealed.

We design a measurement on studying the performance characteristics of scale-out FPGA cloud platforms. We utilize an open-source framework, i.e., InAccel [9], for distributed computing seamlessly integrated with cloud FPGA accelerators. Spark-based applications are fully compatible to the framework and the workloads can be offloaded to the hardware accelerators using InAccel's libraries. Taking advantages of InAccel, we deploy the aforementioned four applications, i.e., digital recognition, optical flow, spam filtering and face detection, on AWS F1 to disclose the scale-out performance of FPGA clouds. We perform the experiments by increasing the number of FPGA boards from 1 to 2 to 4 to 8 Nodes, where the number of nodes is the number of FPGA boards connected to a computing instance. The baseline is the 1 Node case.

Resource Balancing in FPGA IaaS. Figure 3 demonstrates the results of speedup comparison over CPU among 1 Node (baseline), 2 Nodes, 4 Nodes, and 8 Nodes. Taking 2 Nodes experiment as an example, the speedup metric is calculated by dividing the performance of 2 nodes FPGA system by the performance of 2 nodes CPU system, finally divides the baseline case. From the figure, we can notice that the digital recognition performs 1.5×, 2.2×, and 3.0× faster when the number of FPGA instance nodes increases from 1 to 8. The

spam filtering sees 1.1×, 1.2×, and 1.3× speedup for the same increase of FPGA instance nodes. The optical flow application performs similarly to the digital recognition. While the performance of spam filtering and face detection shows relatively lower growing momentum as the system scales out. We can conclude that the performance improvement of the two compute-bound applications grows faster compared to the communication-bound applications.

This performance gap is caused by the variation of communication to compute ratio, i.e., communication time dividing computation time. For the compute-intensive applications, since the compute time is much higher than data transfer time, it is effective to add FPGA accelerators to enlarge the compute capacity and reduce the computation time. However, as for the memory-bound applications, bringing in more accelerators leads to increasing number of partitions in RDD and decreasing the processed data in each node. Due to the inefficiency of small-sized jobs, the compute-bound applications outperform the memory-bound applications in a scale-out system.

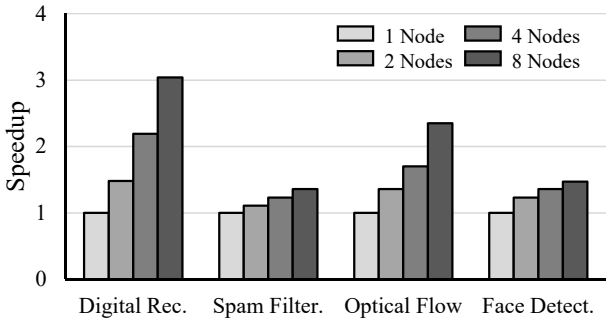

Fig. 3: Speedup over CPU as the FPGA node increases.

Discussion. From the results, we identify two types of resources that affect the performance of FPGA cloud most, i.e., the number of compute units in an FPGA and the communication bandwidth. We point out that the resource balancing in FPGA IaaS will benefit the performance of a scale-out system consisting of FPGA instances. To boost the performance of multi-node accelerator system with memory-bound workloads, increasing the communication bandwidth between FPGA and host should be also taken into account.

Based on the above analysis, we conduct our work to offer a fine-grained approach to understand the real performance of FPGA IaaS.

3.3 Understanding the Performance of FPGA IaaS

To understand the performance of cloud FPGA instances, we need to first review how FPGA IaaS is constructed. A cloud FPGA instance basically packs one cloud compute instance and at least one FPGA board up. For instance, the architecture of AWS F1 is of one cloud server interconnecting with eight FPGAs. A tenant is allowed to rent an FPGA instance with one or more FPGAs (at most eight) according to the requirements for computing capacity of accelerators. The accelerators offered by FPGA IaaS are ready to use for tenants once the FPGA instance is initialized. Each FPGA board is a PCIe device of the cloud compute instance, where

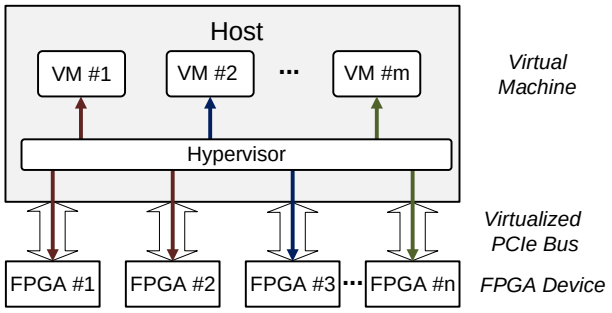

Fig. 4: Architecture of cloud FPGA instance.

the cloud tenant can access the FPGAs without instantiating. From the perspective of CSPs, the FPGAs are provisioned to tenants based on a series of virtualization techniques (e.g., PCIe virtualization) and cloud stacks (e.g., OpenStack in open source community), which is transparent to tenants.

Interestingly, the compute instances are provisioned based on virtualized infrastructure, while the FPGA devices can only be exclusively accessed by one compute instance. As shown in Figure 4, VMs co-locate in one host and share all the physical resources of the server, except the FPGA devices. A compute instance has exclusive access to certain FPGA via the virtualized PCIe bus. Hence, the FPGA IaaS can be broken into three parts: virtual machine, virtualized PCIe bus, and FPGA device.

By evaluating the performance of the three components and identifying the corresponding overhead, we can have a better understanding of FPGA IaaS.

3.3.1 Virtual Machine

Analysis of cloud VMs' performance is a well-studied topic. Existing literature is dedicated to analyzing performance of CPU, storage, networking, and WAN [28], [32], [37]. As for cloud FPGA instances, we mainly focus on device drivers in a VM which enable communications with the FPGA board. Since device drivers define how data is transferred between a VM and FPGA boards, they will significantly impact on the communication efficiency when accelerating various applications with FPGA boards.

3.3.2 Virtualized PCIe Bus

PCIe virtualization technology enables I/O interconnection between a virtual machine and an FPGA device. There exist four state-of-the-art I/O virtualization models supported by virtualization services vendors, i.e., device emulation, para-virtualization, passthrough, and single root input/output virtualization [11]. Compared with other PCIe virtualization techniques, device passthrough assigns PCIe devices directly to a VM bypassing the intervention of host hypervisor, which significantly boosts the performance of data transfer. Moreover, it also enables that guest to exclusively access to the device.

- PCIe Virtualization Overhead. Since only passthrough technique ensures exclusive control of the attached device, we speculate that it is leveraged to virtualize PCIe in FPGA clouds. As a result, we plan to investigate how much overhead the passthrough technique introduces.

- Bandwidth Competition. Additionally, though FPGAs are not shared among VMs in current FPGA IaaS, the PCIe bandwidth may be shared among VMs co-located in the same server. Considering that preemptive bus bandwidth allocation can diminish the performance gained from FPGA accelerator, we need to explore whether PCIe bandwidth competition exists in FPGA clouds, so as to help tenants identify the bottleneck of I/O performance.

3.3.3 FPGA Device

An FPGA board mainly consists of FPGA chip and off-chip memory, i.e., DDR memory. Modern FPGA chips not only contain configurable logic blocks, i.e., lookup tables (LUTs) for general-purpose programmability, but also provide dedicated hard blocks, e.g., digital signal processor (DSP) for special functions. Specifically, LUTs can be reprogrammed to combinations of logic gates; DSP is designed to compute addition and multiplication arithmetic. Benefiting from the parallel "hard" circuits, FPGA is widely used to accelerate various applications.

4 METHODOLOGY

Based on the analysis in Section 3, the performance of cloud FPGA instances is mainly determined by factors from three hierarchy levels. So in this section, we characterize the performance of FPGA IaaS in detail. Specifically, we further identify the factors that affect performance the most in every level. Based on the analysis of the critical components of FPGA IaaS clouds, we propose a fine-grained approach to evaluate FPGA IaaS.

4.1 Bottleneck of Data Transfer in VMs

In Section 3.3.1, we explain the necessity of exploring the device drivers that enable data communication between the VM and FPGA boards. In practice, various factors impact inter-communication efficiency in FPGA clouds, including PCIe device drivers, caching, input/output memory management unit (IOMMU), and non-uniform memory access (NUMA) [33].

However, in the context of cloud FPGA instances, we just need to consider PCIe device drivers, since IOMMU is a configuration setting for a physical server, which is out of tenants' authority. Moreover, there is only one NUMA node as the default setting for cloud VMs, so all memory accesses are local while remote memory access does not exist at all. Finally, since CPU cache is transparent to tenants, we will not evaluate the caching impacts.

With respect to device drivers, we then discuss the available access patterns of PCIe device through drivers.

Access Patterns of PCIe Device. In general, a PCIe device can be accessed using memory map or DMA. Under memory map mechanism, data is transferred via the CPU's address bus. DMA is a hardware mechanism that allows a device to access host memory bypassing CPU. For DMA drivers, which defines data exchange of memory, can be classified as follows:

- Polling v.s. Interrupt. Host either repeatedly polls a status register or waits for the device to send an interrupt request to check whether a read/write transaction has finished.
- Kernel Space v.s. User Space. Typically, kernel space driver uses standard system calls, such as `pwrite` and `pread` to access device memory. As for user space driver, device memory is directly mapped into a block of memory address in the user space.

In [33], the authors evaluate two types of PCIe device drivers, i.e., Linux kernel driver and DPDK driver [10] in modern NIC applications. Compared to [33], our measurement is devoted to understanding the performance of PCIe drivers for FPGA in IaaS clouds. Basically, the methodologies are almost the same, i.e., by measuring the latency of data transmission between hosts and PCIe devices. However, the results of performance are incomparable, because one is for NIC and the other is for FPGA devices. Nevertheless, results of [33] can provide some hints about what factors may impact PCIe driver performance. For exzample, comparing to kernel drivers, we see differences in DPDK that it is implemented with a user-space I/O (UIO) driver while equipped with dedicated CPU core(s) for polling. It means that with polling and UIO optimization, DPDK driver achieves 5% higher throughput than typical kernel driver.

Different from [33], in a cloud environment, things like a VM may incur an extra virtual interrupt thus widening the performance gap between polling and interrupt drivers. Thereby, we conduct measurements to figure out whether the additional performance penalty exists in cloud FPGA instances. Interestingly, we see a 35% gap between polling and interrupt drivers, which is much higher than 5% mentioned above in a bare-metal environment. And the detailed results are demonstrated in Section 5.

4.2 Virtualization Overhead of Intercommunication

As aforementioned in Section 3.3.2, although PCIe passthrough technique can decrease data transfer latency, the overhead is still generated due to traversing multiple layers of virtualization. As a result, concerning the frequent data transfer between VM and FPGA, it is necessary to shed light on the PCIe virtualization overhead in FPGA IaaS. Existing literature already identifies the PCIe virtualization overhead in virtual NICs.

In [43], the authors evaluate the performance of receiving and sending data packets under KVM hypervisor environment. It is reported that with PCIe virtualization, a 10% and 33.2% throughput penalty is seen in the case of receiving and transmitting data packets under KVM hypervisor environment. Different from [43], our measurement studies the PCIe virtualization overhead and bandwidth competition in FPGA IaaS, where many implementation details are transparent and limited programming interfaces are provided. We then conduct a measurement study on investigating the virtualization overhead in FPGA clouds and discussing the impacts on the performance of intercommunication between VM and FPGA.

4.3 Benchmark for FPGA Board

We choose three benchmarks which are frequently used for FPGA acceleration, i.e., single precision general matrix multiplication (SGEMM), advanced encryption standard (AES) and fast fourier transform (FFT), three representative types

of applications demanding for different resources of an FPGA.

- SGEMM. SGEMM is a fundamental operation which occupies almost 95% time of a deep learning training model [25]. In [39], the authors present an FPGA-based deep neural network (DNN) model using SGEMM to perform convolutions. As aforementioned that DSP is dedicated to performing the floating-point multiplication arithmetic, we infer that the number of DSPs will be the bottleneck in an FPGA-based deep learning application.
- AES. AES is an encryption algorithm that is widely available on CPU processors. The encryption process involves bunches of bitwise operations, making it efficient to be accelerated on an FPGA. Much literature has discussed its hardware acceleration solutions with FPGA [23].
- FFT. FTT is widely used to perform convolutional neural network (CNN) models [36]. The implementation of FFT requires a high DDR bandwidth to promise an efficient channel between FPGA and host. Therefore we are eager to know whether DDR will be the bottleneck in an FFT-based CNN model.

TABLE 2: Benchmark for FPGA board.

Benchmark	Benchmark Domain	Benchmark Classification	Crucial Resource
SGEMM	DNN	FP[1] Arithmetic Intensive	DSP
AES	Encryption	Bitwise Operation Intensive	LUTs
FFT	CNN	Memory Intensive	DDR

[1] FP = floating point

The performance of the above benchmarks is mostly determined by a certain type of resources on an FPGA, which we refer to as crucial resources. Table 2 lists our benchmarks and the speculation about corresponding crucial resources. We then try to confirm whether our speculation is valid and help tenants to identify the crucial resource constraints for their applications.

4.4 Metrics

The metrics we select to measure the performance of FPGA IaaS are as follows.

4.4.1 Metric for Data Transfer

- PCIe Throughput. We use PCIe throughput to evaluate the data transfer efficiency between VM and FPGA. The throughput is calculated by dividing the transfer size by the transfer time. The data transfer time is measured by taking a timestamp before issuing packet transactions and taking another timestamp at the end.
- PCIe Latency. We use PCIe latency to evaluate the data transfer speed in a single way. In this case, the PCIe latency is equal to the data transfer time.

4.4.2 Metric for FPGA Board

- Throughput. We take data traffic which can be processed by FPGA per second to evaluate FPGA acceleration efficiency. It is calculated by data traffic size divided by execution time during acceleration on FPGA board.
- Latency. The latency represents the time delay of offloading jobs to external accelerators. It is the sum of FPGA processing time and the data transfer time.

4.5 Measurement Setup

Cloud FPGA Instance Specification. We rent five types of cloud FPGA instances from AWS, the biggest cloud provider in the world, Alibaba Cloud and Huawei Cloud, the biggest cloud providers in China. The instances are AWS F1.2xlarge (AWS F1 for short), Alibaba Cloud ECS.f1-c8f1.2xlarge, ECS.f2-c8f1.2xlarge and ECS.f3-c8f1.xlarge (Ali F1, Ali F2 and Ali F3 for short, respectively) and Huawei Cloud fp1c.2xlarge.11 (HW F1 for short). Table 1 shows the hardware configuration of FPGA instances of Amazon EC2, Alibaba and Huawei cloud.

Amazon FPGA Instance Environment. AWS provides an FPGA Developer AMI for its cloud FPGA instance, where users can easily develop and deploy the FPGA acceleration applications. The AMI is a virtual machine image installed with an operating system as well as requisite services. In our experiments, we use the FPGA Developer AMI-1.4.0 basing on 64-bit CentOS 7.3 operating system. Besides, the FPGA Developer AMI also contains necessary software, simulation interfaces, high-level programming runtime environments and the PCIe DMA driver.

Alibaba FPGA Instance Environment. Similarly, Alibaba Cloud also offers operating system images that contain requisite tools, software and licenses. Here we conduct a series of measurements on three types of Alibaba FPGA instances, i.e., Ali F1, Ali F2 and Ali F3. For Ali F1 instances, they are booted with FaaS F1 Image V1.1_drop1_drop15. As for Ali F2 instances, they are booted with FaaS F2 Image V2.4. And Ali F3 instances are booted with FaaS_F30010_0218 Image. All the VM images aforementioned are basing on 64-bit CentOS 7.3.

Huawei FPGA Instance Environment. Huawei Cloud equips HW F1 instances with a CentOS 7.3 64-bit Image with pre-installed software for FPGA development. The detailed information of developing environment across the five FPGA instances is listed in Table 3.

Method of Measuring Latency. We prepare a string of characters which is randomly generated for each round of data transaction. Before each transaction begins, we use `clock_gettime` system call which provides nanosecond resolution to get start time. After writing data to FPGA, we invoke `fsync` to make sure all the data has been transmitted to the FPGA board. Then we use another `clock_gettime` system call to get end time. PCIe latency is calculated by subtracting the start time from the end time. For each set of data, the experiment is repeated 10 times and average latency is calculated as a final result.

5 Driver Efficiency in Virtual Machine

FPGA clouds are easy for programming owing to a set of mature programming interfaces including the data transfer interface with FPGA device. Taking DMA transaction as an example, for a cloud tenant, the interfaces mask all details

TABLE 3: FPGA Image and OS.

	FPGA Image	OS
AWS F1	AMI-1.4.0[1]	64-bit Centos 7.3
Ali F1	FaaS F1 Image V1.1_drop1_drop15[2]	64-bit Centos 7.3
Ali F2	FaaS F2 Image V2.4[3]	64-bit Centos 7.3
Ali F3	FaaS_F30010_0218[4]	64-bit Centos 7.3
HW F1	CentOS 7.3 64-bit with sdx[5]	64-bit Centos 7.3

[1] https://aws.amazon.com/marketplace/pp/B06VVYBLZZ
[2] https://market.aliyun.com/products/57742013/cmjj022538.html
[3] https://market.aliyun.com/products/57742013/cmjj022540.html
[4] The image is a shared image from Alibaba Cloud
[5] The image id is 0328c25e-c840-4496-81ac-c4e01b214b1f

TABLE 4: Available DMA IP, drivers and interfaces.

	DMA IP	DMA Driver	DMA Interfaces
AWS F1	Xilinx DMA IP	Kernel Space	`pwrite/pread` or `lseek+write/read`
Ali F1	Intel DMA IP	User Space	`fpgaDmaTransferSync`
Ali F2	Xilinx DMA IP	Kernel Space	`pwrite/pread` or `lseek+write/read`
Ali F3	Xilinx DMA IP	Kernel Space	`pwrite/pread` or `lseek+write/read`
HW F1	Xilinx DMA IP	Kernel Space	`pwrite/pread` or `lseek+write/read`

about how DMA engine interacts with kernels and endpoint device while facilitating programming. However, potential performance improvement is also hidden in the high-level programming abstractions. Therefore, in this section, we will disclose the device drivers' impacts on PCIe performance to aid tenants winning maximum performance gains.

Before showing the experimental results, we will describe some necessary measurement setups. We set identical parameters for PCIe configuration, where max payload size is 128 bytes and max read request size is 128 bytes. For all experiments, we carefully control the cache being warmed before the test.

5.1 PCIe Performance of DMA Drivers

Transferring data via DMA basically requires DMA IP cores provided by board vendors, the corresponding DMA drivers and DMA interfaces, as shown in Table 4. Cloud providers will customize the drivers for optimization. For example, Amazon provides two suits of DMA drivers, i.e., XDMA and EDMA, where the latter one is customized for multithread programming. To explore the performance of DMA data transfer, we first evaluate the performance of DMA drivers running in kernel space and user space. Then, for kernel space DMA drivers, we further investigate the performance of XDMA and EDMA under polling and interrupt mode, respectively.

5.1.1 Performance of DMA Drivers in Kernel Space and User Space

Figure 5 and Figure 7 plot the write and read throughput performance of DMA drivers, respectively. The results are normalized to their theoretical throughput. Figure 6 and Figure 8 demonstrate the write and read latency performance of DMA drivers, respectively. As shown in Figure 5, it is obvious that user space driver outperforms kernel space driver remarkably, reaching 60% of its theoretical throughput, while the maximum throughput of kernel space DMA drivers is only no more than 30%. Compared with user space driver, kernel space driver will incur extra overhead due to memory copy between kernel space and user space. Since the memory allocated by the operating system is located at user space while the memory that can be directly accessed by the kernels lies in kernel space, the memory copy operations incur extra performance overhead.

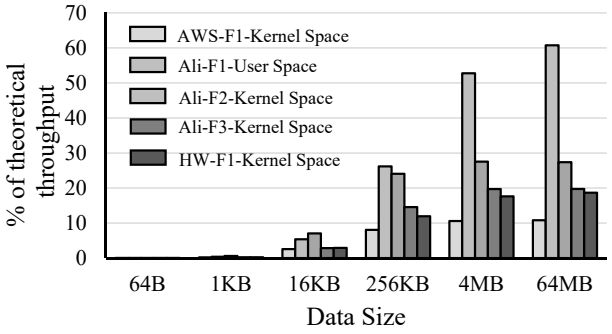

Fig. 5: DMA write throughput of each FPGA instance.

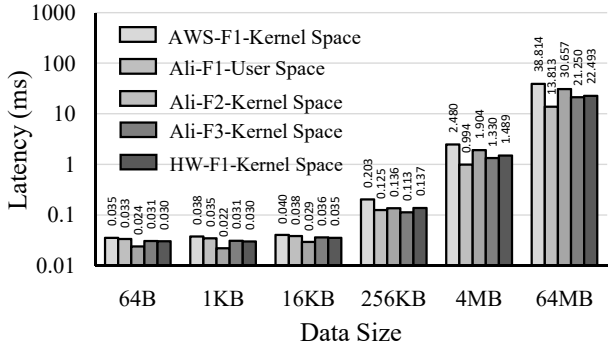

Fig. 6: DMA write latency of each FPGA instance.

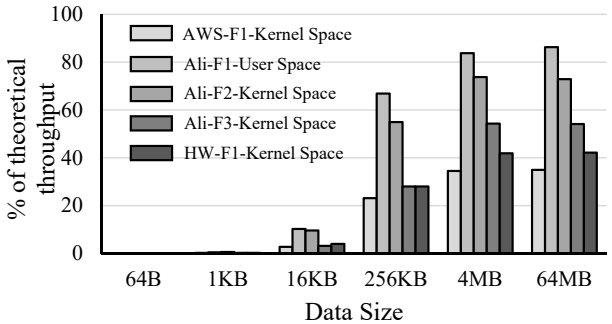

Fig. 7: DMA read throughput of each FPGA instance.

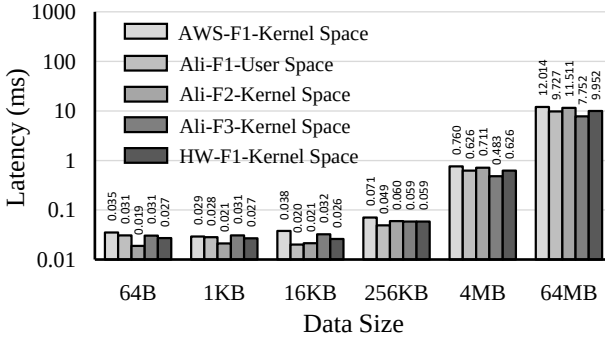

Fig. 8: DMA read latency of each FPGA instance.

In Figure 7, we observe that DMA read performance is consistent with the case of DMA write. User space driver outperforms kernel space driver with a little improvement. Since read performance and write performance present a coherent trend of variation, in the rest of the paper, we only show the write performance results.

As shown in Figure 6 and Figure 8, we find that when the data size is smaller than 16 KB, PCIe latency is measured roughly in tens of nanoseconds. When the data size exceeds 256 KB, PCIe latency grows fast.

Insight. In general, the user space DMA driver has a noticeable performance improvement compared to the kernel space one. Cloud providers are responsible to optimize the general drivers, so as to maximize the performance gained from tenants. In addition, communication latency, i.e., PCIe latency is significant as data size grows large. Cloud providers also need to reduce PCIe latency so that FPGA instances can meet latency-sensitive scenarios, such as networking applications.

5.1.2 Performance of DMA Drivers under Polling and Interrupt Mode.

Figure 9 and Figure 10 plot the PCIe DMA write throughput and write latency of AWS F1 driven by XDMA and EDMA drivers under polling mode and interrupt mode. EDMA driver is customized for multithread programming provided by AWS F1. Overall, we can find that DMA drivers in polling modes are always better than interrupt modes. It is because that the DMA interrupt event occurs in kernel space. CPU has to switch context between the interrupt event and the calling event in interrupt mode, incurring extra overhead. In addition, the performance gap between pooling mode DMA and interrupt mode DMA is as high as 1.35× which is much larger than that in a bare-metal environment (just less than 5%) [33]. We suspect the reason is that extra virtual interrupt controlled by virtual machine manager incurs more overheads. Although polling mode drivers outperform the interrupt mode driver, they are at the cost of consuming significant CPU resources. So targeting compute-intensive applications, it will be a great limitation.

Furthermore, under polling mode, EDMA driver outperforms XDMA significantly (almost 1.6 times higher). In the case of interrupt mode driver, the increase of EDMA write performance is less than 15%. EDMA simplifies the DMA driver by pruning off inactive DMA functionality and only reserving the data transaction functionality[5]. Therefore,

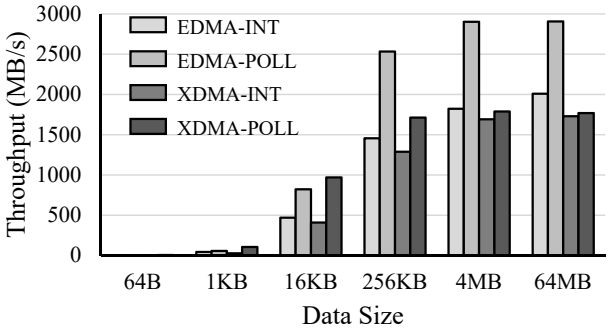

Fig. 9: DMA write throughput of AWS F1 instance driven by XDMA and EDMA under polling and interrupt mode.

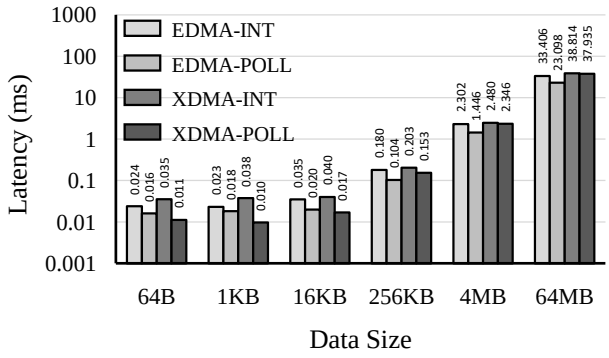

Fig. 10: DMA write latency of AWS F1 instance driven by XDMA and EDMA under polling and interrupt mode.

EDMA can achieve better performance than XDMA. However, the inherent context switch overhead of interrupt mode driver makes it hard to have further improvement.

We also offer a brief view of AWS F1 DMA performance when FPGA Images upgrade. As shown in Figure 11, the write throughput of PCIe DMA varies as the EDMA driver evolves. We note that the performance is improved as the latest DMA drivers that relax the DMA timeouts under interrupt mode.

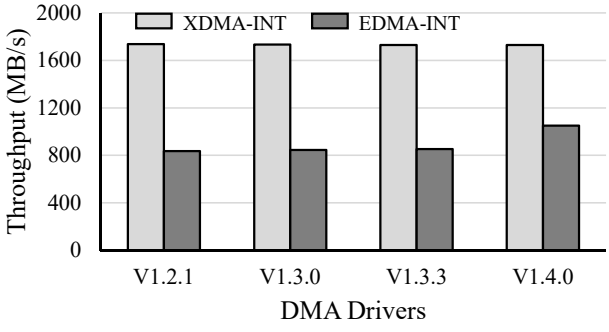

Fig. 11: PCIe DMA write throughput of AWS F1 instance driven by different versions of EDMA and XDMA drivers.

Insight. From the data shown above, optimization of DMA drivers significantly improve the data transfer performance, e.g., the EDMA drive prunes off seldom-used DMA fea-

5. XDMA is an officially provided DMA driver that contains complete functionality, including the basic data transaction, inactive bypass mode, debugging over PCIe and etc. More details can be found in [6].

tures, and simplify the data structure which harvests about 60% throughput increasing. In addition, the DMA drivers provisioned by cloud providers still have room for further improvement. Moreover, since the the mechanism of device drivers is so complicated for tenants to fully understand and make appropriate choices, we highly recommend cloud providers to offer a tuning tool for device drivers that can adapt user applications.

5.2 PCIe Performance of Memory Map

Apart from DMA, an alternative data transfer mechanism is memory map. The memory of a PCIe device is mapped into a memory map region. If the host initiates a memory read request to the PCIe device, it will fetch the data from the memory map region which corresponds to the physical memory address of the PCIe device.

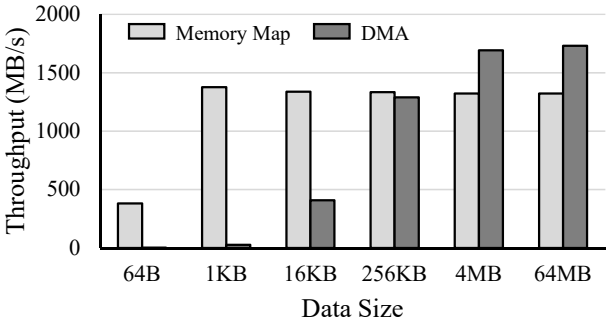

Fig. 12: PCIe write throughput of DMA and memory map.

Figure 12 demonstrates the PCIe write throughput of AWS F1 under DMA and memory map. The DMA driver here refers to XDMA driver under interrupt mode. As the data transfer size varies from 64B to 64 MB, we can observe that the throughput of memory map first increases from 400 MB/s to around 1,300 MB/s then remains steady. DMA throughput is much lower when dealing with small-sized data (less than 16 KB). When transfer data size is bigger, DMA presents a slightly higher performance.

The throughput of memory map remains unchanged when data size is bigger than 1 KB because CPU limits its transfer efficiency. Memory map consumes significant CPU resources when the data traffic ramps up, resulting in long cycles waiting for data to arrive or writing to memory.

Compared to DMA, memory map can be leveraged to transfer small-sized data occasionally, harvesting significant performance gain under small data loads.

6 Virtualization Overhead of Intercommunication

In order to investigate the overhead introduced in cloud FPGA platform, we conduct two experiments on our local server enabling and disabling PCIe passthrough to analyze its overhead. The experiment enabling PCIe passthrough is performed in a guest virtual machine to emulate the FPGA cloud environment. Another experiment is performed in a physical server. The FPGA board we use in the local environment is Xilinx XC709 with PCIe Gen3 x8, and the physical server is Dell R740. The DMA IP core and DMA driver in local environment are kept the same as that of Ali F1 instance.

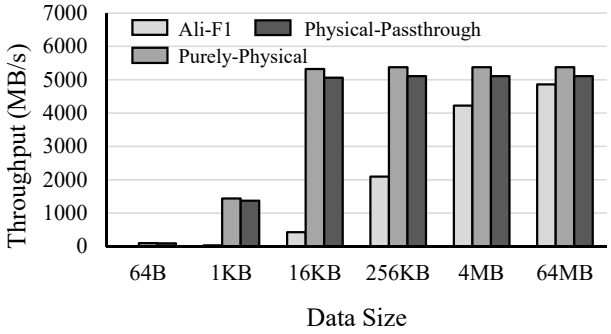

Fig. 13: PCIe write throughput of Ali F1 and local server.

6.1 PCIe Virtualization Overhead

Figure 13 plots the DMA write performance of Ali F1 compared with that in local environment where the FPGA board is directly plugged to the physical server. Purely physical is the experiment without any virtualization. Physical passthrough enables PCIe virtualization and is performed in a VM on our local server. As shown in the figure, DMA throughput reaches the peak when the data size is 16 KB in our local environment. However, the PCIe bandwidth is fully utilized only when the data size is as large as 64 MB in the cloud. Furthermore, PCIe passthrough degrades the DMA performance of local server by around 10%, even more depending on the transfer data size. And the DMA performance of Ali F1 (around 4,800 MB/s) is slightly worse than that of local passthrough (i.e., 5,000 MB/s).

6.2 Bandwidth Competition

To further explore whether PCIe bandwidth competition exists in the cloud, we conduct another experiment that simultaneously transmits data through all four channels of DMA on local server and on the cloud FPGA instance, respectively. As shown in Table 5, we can observe that the overall PCIe throughput is same, i.e., 5,200 MB/s in our local environment. In contrast, the total throughput is improved by 2.8× while the PCIe throughput degrades 30% for each channel.

TABLE 5: PCIe DMA write throughput when transmitting data through one channel and four channels.

	Cloud (MB/s)		Local (MB/s)	
	Each	Total	Each	Total
1x Channel	1000	1000	5200	5200
4x Channels	700	2800	1300	5200

Since we cannot restore the same virtualization strategy as FPGA cloud leverages, in this part, we just prove the existence of PCIe virtualization overhead in cloud. From our experiment, the local server suffers from a 10% of performance penalty, we can suspect a similar penalty is also experienced by the FPGA clouds. Besides, we identify PCIe bandwidth competition will degrade considerable I/O performance of individual tenant.

Insight. Based on the above measurements (including Section 5), compared with FPGA which is directly plugged

in the physical server, I/O performance of cloud FPGA instances significantly degrades to half its theoretical bandwidth due to the overhead introduced by a series of software, including drivers, hypervisor, and so on. As a result, it is necessary for CSPs to optimize the I/O performance, so as to mitigate the data transfer overhead in cloud. Only when CSPs offer sufficient quality of service, can tenants enjoy the maximum acceleration performance benefits.

7 Performance of FPGA Board

Cloud tenants typically rent cloud FPGA instances to accelerate deep learning, complex data analysis applications and etc. As applications vary, tenants should select appropriate instance types to meet their needs. In this section, we choose three representative FPGA acceleration benchmarks, respectively SGEMM, AES and FFT to evaluate the FPGA instances.

7.1 SGEMM Acceleration

We choose SGEMM with input of floating point matrices to evaluate the performance of FPGA instances running floating-point arithmetic intensive workloads. The standard form of SGEMM is typically defined as follows.

$$\mathbf{C} \leftarrow \alpha\mathbf{AB} + \beta\mathbf{C},$$

where $\mathbf{A} = [a_{ij}]_{N\times K}$, $\mathbf{B} = [b_{ij}]_{K\times M}$, and $\mathbf{C} = [c_{ij}]_{N\times M}$ are dense matrices. Without affecting any results, we simply assume N=K=M in setup.

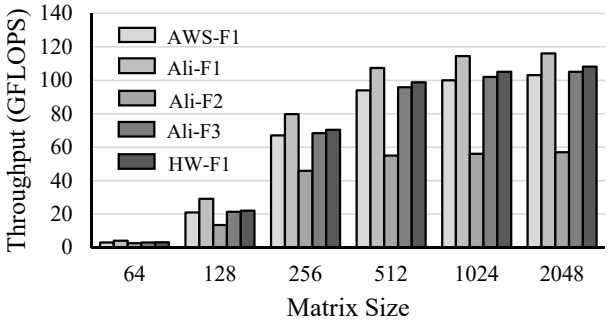

Fig. 14: SGEMM throughput of each cloud FPGA instance.

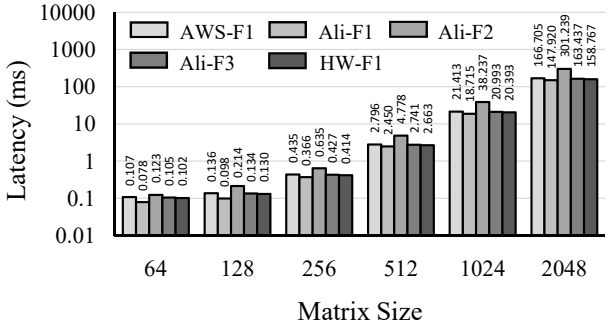

Fig. 15: SGEMM latency of each cloud FPGA instance.

Figure 14 and Figure 15 illustrate the SGEMM throughput and latency across FPGA instances with increasing matrix size. In general, for all the FPGA instances, the throughput increases as the matrix size becomes larger. Specifically, Ali F1 instance outperforms the other four with the peak throughput of 116 GFLOPS. Ali F2 instance produces a steady but worst throughput ranging from 30 to 60 GFLOPS. The principle of SGEMM is decomposing large matrix into small matrices recursively before computation. The block-wise operations allow a fully pipelined implementation in FPGA with remarkable speedups, especially when the matrices are very large. Therefore, the performance is better in the context of larger matrix size. In addition, performing floating point operations especially multiplications consumes lots of DSP resources. Since Ali F1 FPGA provides over one thousand optimized floating-point DSP blocks to support floating point operations, it gains considerable acceleration performance. AWS F1, Ali F3 and HW F1 also equip thousands of DSP slices and achieves comparable throughput, but Ali F2 is limited by its DSP resources, thus it quickly reaches its compute capacity and performs the worst.

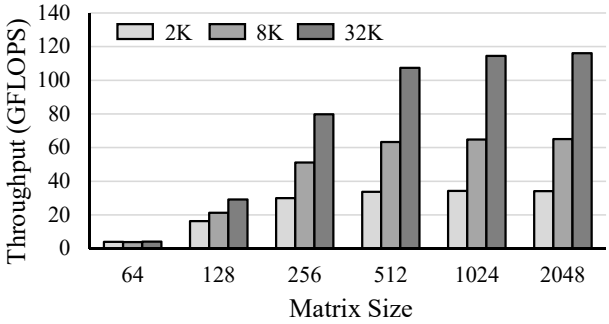

Fig. 16: SGEMM throughput of Ali F1 when allocating different memory sizes on FPGA.

Apart from DSP resources, the performance of SGEMM is sensitive to both temporal and spatial locality of data. Therefore, we conduct another experiment on Ali F1 instance to explore the impacts of allocated memory size on SGEMM throughput performance. As shown in Figure 16, as the matrix size increases to 1024x1024, the throughput reaches the peak value of 100 GFLOPS when allocating enough memory size (32 KB). On the contrary, as the size of local memory decreases, the SGEMM throughput gets lower. Because input matrices are frequently read, allocating adequate memory for buffering the data locally can significantly reduce miss rates and further decrease the total execution time.

Insight. DSP resources are ideal choices for floating point operations. We take SGEMM as one example to advise tenants to focus on the DSP slices if their applications involve a large volume of floating point operations.

7.2 AES Acceleration

AES is an encryption algorithm that requires a large amount of LUTs since it involves a bunch of logic gate operations and shift operations. In this experiment, we conduct an AES-128 ECB decryption task on FPGA and verify the results on CPU.

As plotted in Figure 17 and Figure 18, we observe that for AES encryption, the performance of AWS F1, Ali F2, Ali

F3 and HW F1 instances are comparable, which are all based on Xilinx FPGA chips, outperforming that of Ali F1, which is based on Intel FPGA chips. As the encryption algorithm barely buffers data, the performance does not change much as the dataset size grows larger. We can easily understand that Xilinx FPGA instances have bare performance differences since they share the same FPGA architecture.

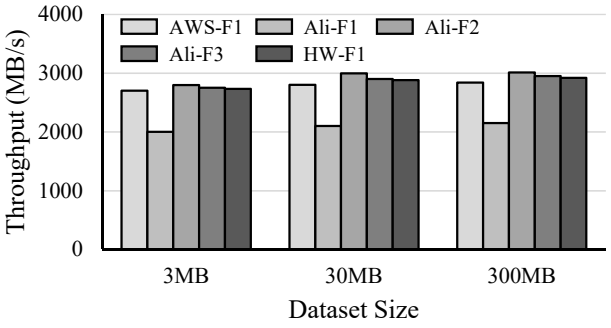

Fig. 17: AES throughput of each cloud FPGA instance.

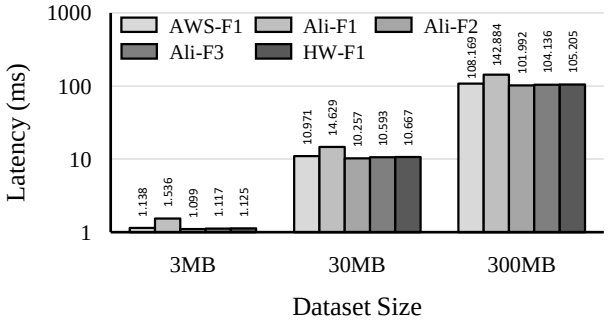

Fig. 18: AES latency of each cloud FPGA instance.

Insight. We take AES as an example to illustrate that dealing with compute-intensive workloads which involve bunches of logic gates operations. Tenants can focus on the capacity of the logic cells, provided by each FPGA board when accelerating complex applications.

7.3 FFT Acceleration

FFT is a mathematical method that changes the signals from time domain to frequency domain to analyze its frequency characteristics. Researchers pursue deploying FFT in deep learning applications to train CNN model leveraging its Fourier convolutions features. Table 6 shows that dealing with large volume of data, AWS F1, Ali F3 and HW F1 instances can achieve higher throughput and lower latency compared with the other instances.

DDR Bandwidth. In order to verify the above results, we deploy a measurement to claim the DDR bandwidth. To reduce the impacts of PCIe latency, we pipeline the reading and writing to reduce the number of transfers. To make it clear, if the transfer size is 64 MB, the number of blocks is 64, and then for each block, the data size is 1 MB. We simultaneously read/write 64 blocks of data, and the transfer time is reduced to 1/64 compared with no pipeline. In general, the bandwidth of DDR is proportional to its number of banks, so in our experiment, we use the least one, i.e., two DDR banks bandwidth to demonstrate their performance. Table 7 lists the DDR configurations.

TABLE 6: FFT throughput and latency.

	FFT Length	Throughput	Latency
AWS F1	1024	5500 MB/s	85.0ms
Ali F1	1024	3640 MB/s	128.4ms
Ali F2	1024	3860 MB/s	121.1ms
Ali F3	1024	5420 MB/s	86.2ms
HW F1	1024	5470 MB/s	85.5ms

TABLE 7: DDR specifications.

	Storage Size	Theoretical Bandwidth	Frequency
AWS F1	4x16GB	68 GB/s	2133 MHz
Ali F1	2x8GB	34 GB/s	2133 MHz
Ali F2	4x4GB	76.8 GB/s	2400 MHz
Ali F3	4x16GB	68 GB/s	2133 MHz
HW F1	4x16GB	68 GB/s	2133 MHz

As demonstrated in Figure 19, with the bulk of data growth, the bandwidth of DDR increases significantly. Furthermore, DDR in AWS F1, Ali F3 and HW F1 instances gets a better performance with a bandwidth of over 26 GB/s, much higher than that of Ali F1 and Ali F2, whose DDR bandwidth only reach around 5 GB/s and 10 GB/s, respectively. Such results coincide with that in Table 6, where AWS F1, Ali F3 and HW F1 have the highest performance under FFT workloads.

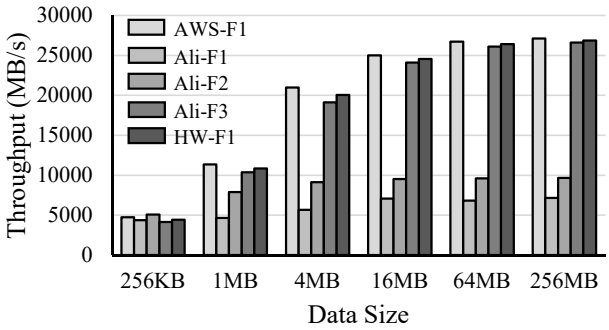

Fig. 19: DDR throughput of each cloud FPGA instance.

Insight. DDR resource plays a critical role in improving the performance of memory-intensive workloads. As a result, cloud tenants should select a cloud FPGA instance with high DDR bandwidth for achieving better throughput.

8 Conclusion

By empirically analyzing the performance of cloud FPGA instances, this paper proposes a fine-grained approach to understand FPGA IaaS. First of all, we evaluate the acceleration performance to provide an overall understanding of FPGA IaaS. Motivated by the observations of the performance gain of single-node and multi-node cloud FPGA platforms, we further explore the performance of cloud

FPGA instances from three aspects, i.e., virtual machine, virtualized PCIe bus, and FPGA device. With respect to virtual machine, we conduct a series of measurements on the performance of two fundamental data transfer mechanisms, i.e., DMA and memory map. As for DMA, we focus on understanding how DMA working mode and DMA drivers impact the I/O performance between the compute instance and FPGA boards in public cloud. For virtualized PCIe bus, we examine the overhead of data transfer between the compute instance and FPGA boards. In terms of FPGA device, we run various benchmarks ranging from compute-bound to memory-bound algorithms, including SGEMM, AES, and FFT, so as to compare how FPGA boards in FPGA IaaS perform when accelerating various workloads.

Our study produces multiple key takeaways:

1) Concerning the performance-cost imbalance in FPGA IaaS, we highly suggest that tenants select low-end FPGA instances for high cost-efficiency as much as possible, unless their applications strictly require high performance, e.g., low latency or high throughput.
2) Striking a balance between data processing speed and data transfer speed in FPGA IaaS will benefit the performance of FPGA clouds for various applications. We suggest that tenants take both factors into account in their system designs.
3) Since general DMA drivers provisioned by FPGA providers require further optimization, tenants need to choose CSPs who offer customized DMA drivers to obtain better performance. In the meantime, CSPs are obligated to improve driver performance to adapt their cloud platforms.
4) We identify the PCIe overhead of data transfer between the compute instance and FPGA boards. Tenants need not to accelerate traffic-intensive workloads in FPGA IaaS and CSPs are responsible to mitigate the overhead to meet tenants' requirements.
5) For different types of workloads, the acceleration performance is typically determined by one critical kind of FPGA resources. Tenants can select a proper cloud FPGA instance based on our results.

REFERENCES

[1] Alibaba Cloud FPGA as a Service Instances. https://www.aliyun.com/product/ecs/fpga/.
[2] Alibaba Cloud FPGA as a Service Instances. https://cloud.tencent.com/product/fpga.
[3] Amazon EC2 F1 Instances. https://aws.amazon.com/cn/ec2/instance-types/f1/.
[4] Apache SpamAssassin Dataset. https://spamassassin.apache.org/old/publiccorpus/.
[5] Azure Machine Learning Service Documentation. https://docs.microsoft.com/en-us/azure/machine-learning/service/.
[6] DMA/Bridge Subsystem for PCI Express v4.1 Product Guide. https://www.xilinx.com/support/documentation/ip_documentation/xdma/v2_0/pg195-pcie-dma.pdf.
[7] For Intel, Adding Altera, FPGA Hardware Is Easy: Next Comes Supporting Software. https://www.forbes.com/sites/kurtmarko/2015/06/09/intel-fpga-software/.
[8] FPGA-accelerated Cloud Server. https://www.huaweicloud.com/en-us/product/fcs.html.
[9] Inaccel - Accelerated Solutions for the Cloud. https://www.inaccel.com/.
[10] Intel DPDK. https://dpdk.org/.
[11] Linux Virtualization and PCI Passthrough. https://www.ibm.com/developerworks/linux/library/l-pci-passthrough/.
[12] MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proc. of OSDI*, 2016.
[14] Y. M. Alkabani, M. W. El-kharashi, and H. S. Bedor, "Hardware/software partitioning of a bayesian spam filter via hardware profiling," in *Proc. of IEEE International Symposium on Industrial Electronics*, 2006.
[15] M. Asiatici, N. George, K. Vipin, S. A. Fahmy, and P. Ienne, "Virtualized execution runtime for fpga accelerators in the cloud," *IEEE Access*, 2017.
[16] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. of European Conference on Computer Vision*, 2012.
[17] S. Byma, J. G. Steffan, H. Bannazadeh, A. Leon-Garcia, and P. Chow, "Fpgas in the cloud: Booting virtualized hardware accelerators with openstack," in *Proc. of IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 2014.
[18] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, "A cloud-scale acceleration architecture," in *Proc. of MICRO*, 2016.
[19] F. Chen, Y. Shan, Y. Zhang, Y. Wang, H. Franke, X. Chang, and K. Wang, "Enabling fpgas in the cloud," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, 2014.
[20] Y.-T. Chen, J. Cong, Z. Fang, J. Lei, and P. Wei, "When spark meets fpgas: A case study for next-generation DNA sequencing acceleration," in *Proc. of HotCloud*, 2016.
[21] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M. Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg, "Azure accelerated networking: Smartnics in the public cloud," in *Proc. of NSDI*, 2018.
[22] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications," in *Proc. of FPGA*, 2012.
[23] T. Good and M. Benaissa, "Aes on fpga from the fastest to the smallest," in *Proc. of 7th International Workshop of Cryptographic Hardware and Embedded Systems*, 2005.
[24] M. Huang, D. Wu, C. H. Yu, Z. Fang, M. Interlandi, T. Condie, and J. Cong, "Programming and runtime support to blaze fpga accelerator deployment at datacenter scale," in *Proc. of ACM Symposium on Cloud Computing*, 2016.
[25] Y. Jia, "Learning semantic image representations at a large scale," Ph.D. dissertation, UC Berkeley, 2014.
[26] S.-W. Jun, M. Liu, K. E. Fleming, and Arvind, "Scalable multi-access flash store for big data analytics," in *Proc. of FPGA*, 2014.
[27] S.-W. Jun, M. Liu, S. Lee, J. Hicks, J. Ankcorn, M. King, S. Xu, and Arvind, "Bluedbm: An appliance for big data analytics," in *Proc. of ISCA*, 2015.
[28] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: Comparing public cloud providers," in *Proc. of IMC*, 2010.
[29] X. Li, X. Wang, F. Liu, and H. Xu, "Dhl: Enabling flexible software network functions with fpga acceleration," in *Proc. of ICDCS*, July 2018.
[30] J. W. Lockwood and M. Monga, "Implementing ultra low latency data center services with programmable logic," in *Proc. of IEEE Annual Symposium on High-Performance Interconnects*, 2015.
[31] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J. Kim, and H. Esmaeilzadeh, "Tabla: A unified template-based framework for accelerating statistical machine learning," in *Proc. of HPCA*, 2016.
[32] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *Proc. of ICC*, 2012.
[33] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore, "Understanding pcie performance for end host networking," in *Proc. of SIGCOMM*, 2018.
[34] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. O. G. Hock, Y. T. Liew, K. Srivatsan, D. J. M. Moss, S. Subhaschandra,

and G. Boudoukh, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proc. of FPGA*, 2017.
[35] M. J. H. Pantho, F. Hategekimana, and C. Bobda, "A system on fpga for fast handwritten digit recognition in embedded smart cameras," in *Proc. of ICDSC*, 2017.
[36] H. Pratt, B. Williams, F. Coenen, e. M. Zheng, Yalin", J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, "Fcnn: Fourier convolutional neural networks," in *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2017, pp. 786–798.
[37] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of i/o workload in virtualized cloud environments," in *Proc. of ICC*, 2010.
[38] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J. Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proc. of ISCA*, June 2014.
[39] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," *CoRR*, vol. abs/1610.00324, 2016. [Online]. Available: http://arxiv.org/abs/1610.00324
[40] Viola, Jones, and Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 734–741 vol.2.
[41] J. Weerasinghe, R. Polig, F. Abel, and C. Hagleitner, "Network-attached fpgas for data center applications," in *Proc. of International Conference on Field-Programmable Technology (FPT)*, Dec 2016.
[42] Z. Wei, L. Dah-Jye, and B. Nelson, "Fpga-based real-time optical flow algorithm design and implementation," *Journal of Multimedia*, vol. 2, 09 2007.
[43] J. F. Zazo, S. Lopez-Buedo, Y. Audzevich, and A. W. Moore, "A pcie dma engine to support the virtualization of 40 gbps fpga-accelerated network appliances," in *Proc. of International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2015.
[44] Y. Zhou, U. Gupta, S. Dai, R. Zhao, N. Srivastava, H. Jin, J. Featherston, Y.-H. Lai, G. Liu, G. A. Velasquez, W. Wang, and Z. Zhang, "Rosetta: A realistic high-level synthesis benchmark suite for software programmable fpgas," in *Proc. of FPGA*, 2018.

Xiuxiu Wang received her B.Eng. degree in the School of Optical and Electronic Information, Huazhong University of Science and Technology, China. She is currently a M.Eng. student in the School of Computer Science and Technology, Huazhong University of Science and Technology, China. Her research interests include cloud computing and FPGA acceleration.

Yipei Niu received his B.Eng. degree from Henan University, and M.Engr. degree from Huazhong University of Science and Technology. He is currently a Ph.D. student in the School of Computer Science and Technology, Huazhong University of Science and Technology, China. His research interests include cloud computing, container networking, serverless computing, and FPGA acceleration.

Fangming Liu received the B.Eng. degree from the Tsinghua University, Beijing, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong. He is currently a Full Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and edge computing, datacenter and green computing, SDN/NFV/5G and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars, and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, as well as the First Class Prize of Natural Science of Ministry of Education in China.

Zichen Xu received his Ph.D. degree from the Ohio State University. He is a full professor in Nanchang University, China. His research spans in the area of data-aware complex system, including profile analysis, system optimization, and data management system design / implementation.