Network Attached FPGAs in the Open Cloud Testbed (OCT)

Suranga Handagala*, Miriam Leeser*, Kalyani Patle†, Michael Zink†
Northeastern University*, University of Massachusetts Amherst†
*s.handagala|m.leeser@northeastern.edu, †kpatle|mzink@umass.edu

***Abstract*—The Open Cloud Testbed (OCT) provides nodes with Field Programmable Gate Arrays (FPGAs) that are under the complete control of the user and are directly attached to a network switch via two 100Gbps connections. We provide TCP and UDP stacks on the FPGAs. In addition, users have the ability to experiment with their own protocol. We present several experiments which make use of this capability including TCP throughput measurements, an encryption/decryption example, and machine learning inference split across two FPGAs where the images are input on one node and the labelled output available on a second node. The testbed is available for researchers to perform their own experiments, and includes a development platform that allows users to create FPGA applications. Network measurement results show we achieve close to peak bandwidth by tuning appropriate parameters.**

***Index Terms*—cloud computing, testbeds, bare metal, FPGA**

I. Introduction

Virtually all of the online services offered today rely on computing and storage that is performed in the cloud. This paradigm has changed the way users share information, consume services, and execute applications. In addition, cloud services have been the foundation for a wide range of new technologies and applications that are made available to a broad user community. An additional driver for innovation is the development and deployment of new hardware in compute clouds, which in turn offers new features for cloud providers.

Many areas of innovation are limited to researchers and developers working for today's public clouds, where they can perform experiments at massive scale, with real users, taking advantage of detailed telemetry from cloud infrastructure, and where they are ultimately able to transition successful innovation into products made available to their customers. Researchers and developers outside these organizations are at a disadvantage since they do not have access to public clouds at the level that would be required to perform systems research. In addition, public clouds typically restrict researchers in terms of new paradigms and topologies and uses of hardware that can be experimented with.

Realizing the critical importance of enabling the systems research community to participate in this fundamental transformation in IT, research agencies have made significant investments to build cloud testbeds (e.g., the National Science Foundation in 2014 and 2017 invested ∼$40M for the development and operation of two cloud testbeds).

Over the past two years, we have been developing the Open Cloud Testbed (OCT) [27], with the goal to build and support a testbed for research and experimentation into new cloud platforms — the underlying software and hardware which provides cloud services to applications. Testbeds such as OCT are critical for enabling research into new cloud technologies – research that requires experiments which potentially change the operation of the cloud itself.

This paper focuses on the integration of Field-Programmable Gate Arrays (FPGA) in OCT. While FPGAs are already available in public clouds (e.g., Azure [26] and Amazon F1 [5]) those services do not offer the low level access and programmability that is required by systems researchers. In OCT, researchers have the ability to allocate bare metal servers that include FPGAs for their experimentation. This allows them to develop and evaluate new cloud computing architectures that use FPGAs as processing units in addition to CPUs and GPUs. OCT allows researchers to experiment with FPGAs directly connected to the network, a topology not exposed to users by existing public clouds. OCT thus supports research in areas such as distributed and cloud computing, disaggregation, sustainable computing, computer networking, and machine learning.

The FPGAs in OCT are connected to a host processor via PCIe, and are also directly connected to a network switch, allowing FPGA-to-FPGA communication via two 100 Gbps Ethernet links as well as connections from an FPGA to other network-attached devices. A feature of OCT is that researchers can experiment with this topology in ways not available from other clouds. FPGAs directly connected to the network bring the advantage of reduced latency, dedicated network interfaces and isolation from the compute load of the host system. In contrast, accelerators typically access a network interface via the host by communicating data over the PCIe bus.

The FPGA interacts with the network via hardware implementations of standard interfaces. In this project we make use of available UDP [25] and TCP/IP [11] implementations. The user application implemented in hardware communicates directly with these components through a control interface. The parameters specified to the hardware are analogous to those used in software and include target IP address, etc. for TCP/IP. In this paper, we study the performance of the network connections as well as how the networking setup of FPGAs impacts the performance of applications that rely on direct FPGA-to-FPGA communication.

The contributions of this paper are:

- Using the direct network connections of the FPGAs in

978-1-6654-0926-1/22/$31.00 ©2022 IEEE

OCT and measuring their performance.

- Splitting a machine learning application across two FPGAs and having them talk to one another such that the inputs are received from the host of one FPGA and outputs processed and received by a host computer connected to a second FPGA. This application makes use of both Ethernet connections available to double the throughput realized.
- Providing a platform that other researchers can build on by implementing their own protocols or higher layers of the network stack based on FPGAs connected to the network using Ethernet.

The rest of the paper is outlined as follows. In Sec. II, we provide background information on FPGAs in the cloud and give an overview of the Open Cloud Testbed. Sec. III presents measurements and applications making use of the network connected FPGAs. We discuss lessons learned and future directions in Sec. IV and conclude in Sec. V.

II. Background and Related Work

A. Open Cloud Testbed

In this section, we describe the current state of the Open Cloud Testbed and its features. We describe the current hardware available for researchers, highlight the newly-deployed FPGA accelerators and their usage, and provide an overview of adjacent resources from which researchers can benefit while conducting experiments. Figure 1 give an overview of the OCT topology and the network infrastructure.

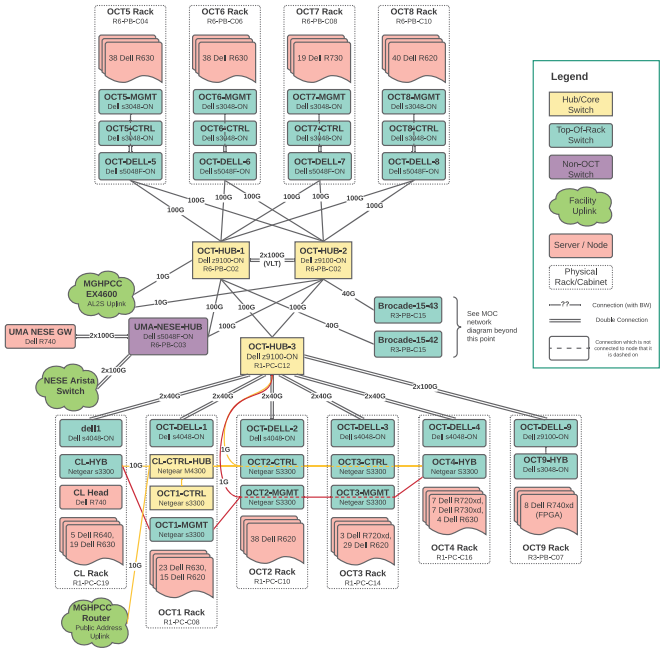

Fig. 1. Overall OCT topology, currently composed of ten racks of compute equipment. Servers are connected via 2x40G or 100G Ethernet connection; each FPGA is connected via 2x100G Ethernet.

1) Hardware: Since the inception of OCT we were able to constantly grow its footprint by adding two very significant increments of hardware. Originally, the testbed started as a small cluster of only five nodes. Based on two rounds of generous donation of equipment from industry partner Two Sigma, donated to the Massachusetts Open Cloud (MOC) and made available to OCT, we were able to increase the size of the testbed significantly. As of December 2021, the testbed consists of a total of 5,172 cores and 63 TB of RAM distributed over 237 servers. Since this hardware has been donated by industry we have the opportunity to make it available to not only the CISE research community, but to select open source communities as well. Through the Elastic Secure Infrastructure (ESI) service [22], the hardware can be made available to different control frameworks.

The operation of a testbed that supports systems research and offers bare metal servers requires three separated networks. First, a 1Gbps management network, providing a private network that connects the management instance with the IPMI interfaces of the servers. Second, a public 1Gbps control network, which allows experimenters to connect to the individual servers via the public internet. Third, a data plane network that establishes high-bandwidth (40Gbps or 100Gbps) connections between the individual nodes. As can be seen in Fig. 1, half of the racks are connected via 2x40Gbps, while the other half are connected via 2x100Gbps connections. (This difference in bandwidth is based on the growth of the testbed infrastructure in two increments.) Finally, the use of VLANs enables the creation of isolated networks, where a separate VLAN is assigned to each experiment.

In the following section we present further details on the FPGAs that are installed in several OCT nodes.

2) FPGAs: At the onset of the OCT project, we surveyed a broad group of researchers from the FPGA community to identify their needs for a testbed that offers such devices. Based on their feedback and our own experience in FPGA-related systems research we integrated FPGAs in OCT and created a set of tools readily usable by experimenters.

We have successfully integrated eight Xilinx Alveo U280s into OCT (and are in the process of adding an additional eight). Alveo U280s are top of the line FPGA accelerator cards for the cloud, with High Bandwidth Memory to support data intensive applications. The FPGAs use PCIe connection to the host processor and are also directly connected to a network switch via two independent 100Gbps connections. This allows direct FPGA-to-FPGA communication, which is currently supported via TCP and UDP stacks. FPGA communication directly to/from other devices is also possible. A researcher can experiment with their own protocol provided it runs on top of Ethernet, which is needed to communicate with the network switch. We refer the interested reader to Leeser et al. [18] for additional details on the FPGA setup in OCT.

The nodes that host the FPGAs are allocated via the CloudLab framework [10]. In addition, we have created a virtual machine image that includes the Xilinx Vitis tools and can be instantiated in MOC (Fig. 2). To support the research

community, we created tutorials for all steps required – from the creation of a bitstream to the execution of an experiment in the testbed [14]. In addition, users can clone the FINN workflow—an experimental framework from Xilinx Research Labs to explore deep neural network inference on FPGAs [8]. Tutorials for running FINN examples have been adapted to the OCT setup and are also available [14]. This toolchain provides researchers with the components necessary to implement an application that can run on FPGAs to evaluating such applications in an actual testbed.

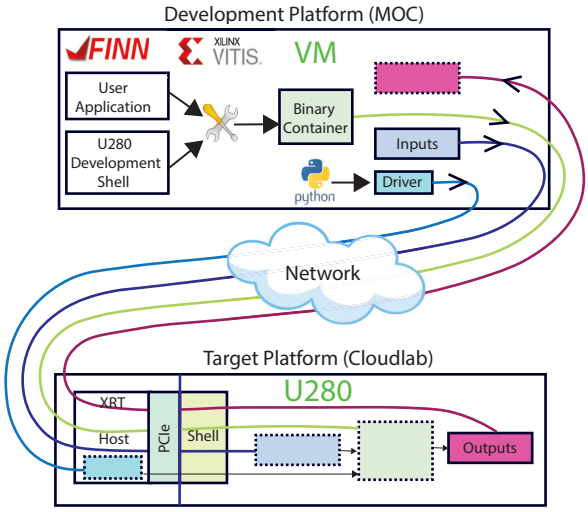

Fig. 2. OCT FPGA tool workflow.

B. Related Work

The standard interface to accelerators such as FPGAs and GPUs is through a host processor via PCIe. This introduces a great deal of latency as data needs to go from network to processor to accelerator for processing and then back to the processor before being transmitted to the network. All of this communication can eliminate any speedup the accelerator may provide for many applications. As a result, many researchers are investigating the direct connection of accelerators to the network or to storage elements in order to remove this overhead.

To support FPGA direct communication to the network, protocols such as UDP or TCP/IP or a custom protocol built on top of Ethernet must be supported. The work presented in this paper builds on the Xilinx UDP stack [25], and the TCP/IP support for FPGAs provided by researchers at ETH Zurich [11], [20]. This group has further proposed a Data Processing Interface (DPI) to take advantage of programmability at many different layers of a network stack [3].

Several research groups are investigating the use of MPI on FPGAs [13], [21]. While this project is not investigating MPI directly, the infrastructure in OCT supports the implementation of such projects.

Microsoft Azure [12] implements a SmartNIC on an FPGA, and makes use of the FPGA as a "bump in the wire" that can process packets as they are being transmitted and received. This infrastructure is used by Microsoft programmers; it is not available for users to program the FPGAs directly. FPGAs that users can program directly are available from AWS [5] and other cloud providers. In many cases the way that these FPGAs can be used is restricted. For example, AWS does not support network attached FPGAs.

Both Xilinx and Intel have systems available for academic researchers. Intel supports Hardware Accelerator Research Program (HARP); the Xilinx Adaptive Compute Cluster (XACC) Program [24] funds centers at ETH Zurich, National University of Singapore, UCLA, and UIUC. These programs make accelerators in advanced configurations available to researchers. However, the scope of these systems is small and they are only available directly through each vendor and not as open resources to the research community.

III. Applications and Results

In this section we present use cases for network attached FPGAs. First we present measurements for FPGA to CPU and FPGA to FPGA direct communication using TCP, then we show a simple example where data is encrypted on an FPGA, transmitted and then decrypted on a second FPGA connected via UDP. Finally we present a machine learning application that makes use of both links from FPGA to network to process data in two different directions.

A. TCP Throughput Measurements

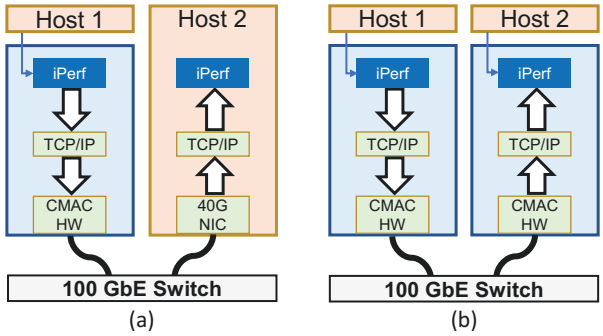

Fig. 3. Throughput measurement setup for (a) FPGA-to-host communication, (b) FPGA-to-FPGA communication.

1) FPGA-to-host: In OCT, each server with a Xilinx Alveo U280 also has a 40 GbE NIC installed, enabling FPGA-to-host experiments. To measure network throughput, we use the TCP/IP stack from ETH Zurich [11] and an iPerf client on the FPGA to communicate with an iPerf server on another host via its 40 GbE interface, as shown in Fig. 3(a). While we have used two separate hosts for this experiment, this is not required; it is also possible to have the iPerf client and the iPerf server on the same physical server. The connection from the FPGA to the host is through the switch, which enables communication between the two.

For this experiment, we ran the iPerf client on the FPGA and changed the packet size from 768 to 1472 Bytes in 64

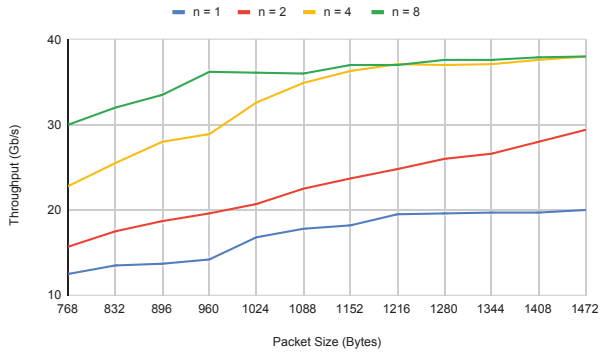

Fig. 4. TCP throughput vs. packet size for FPGA-to-host communication.

Bytes increments. Each iPerf sesion runs for 1 second. The iPerf client also has an option to change the number of TCP connections (n). The results in Fig. 4 show the throughput measurements reported by the iPerf server at different values of n. Note that a single connection could only utilize about one third of the total bandwidth at the lowest packet size, which increases to 50% at the highest size. Increasing n results in much larger bandwidth utilization even at low packet sizes, and the network is saturated when n is increased to 4 or 8. In the future we plan to repeat these experiments with nodes with 100 GbE interfaces.

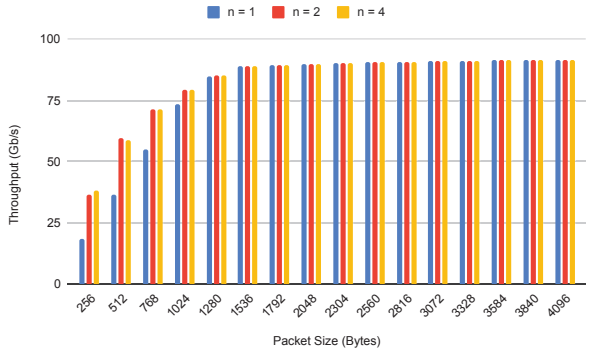

Fig. 5. TCP throughput measurement results for FPGA-to-FPGA communication.

2) FPGA-to-FPGA: The FPGA-to-FPGA throughput measurement experiment, shown in Fig. 3(b), uses the ETH Zurich's TCP/IP stack [20] along with iPerf kernels implemented on both FPGAs to characterize the performance of the network. The iPerf kernel has registers to count the number of packets sent/received and the number of clock cycles taken. We have exposed these registers to the host processor to read them and calculate the throughput on both ends. We have varied the packet size from 256 Bytes to 4096 Bytes and measured the throughput at different values of n; results are shown in Fig. 5. The throughput variation is linear up to about 1280 Bytes when $n = 1$ which eventually gets saturated at 90-91 Gb/s level. As we increase n, note that more bandwidth is utilized even when the packet size is low. Regardless of the value of n, the link saturates when the packet size is increased further.

B. Encryption/Decryption

We have implemented a simple encryption and decryption demo as shown in Fig. 6. Here, the sender side FPGA encrypts the user data read from a text file stored on a host, and transmits the encrypted data over the network. The receiver side FPGA decrypts the data and forwards them back to the sender side FPGA where we have implemented pass-through logic to transfer the message back to the the original host. Thus, we can verify that the transmitted and received messages are the same. We have used the AES encryption/decryption logic from [19] in this implementation.

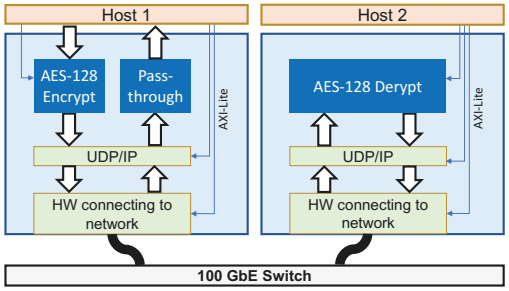

Fig. 6. Encryption and decryption on network attached FPGAs.

C. Distributed DNN Inference

Machine learning inference is a popular application for FPGAs. We demonstrate the capabilities of the network attached FPGAs in OCT by implementing MobileNet [15] using the FINN [8] framework. MobileNet is one of the provided FINN examples [23] and partitioning a FINN graph into multiple FPGAs has been discussed in [4]. This design uses two local accelerators and one split accelerator implemented on two FPGAs. Our implementation differs from the one provided as its input and output channels are on separate FPGAs, and we make use of both network ports as shown in Fig. 7. The MobileNet design is split between two FPGAs using direct communications via the 100GbE switch. We used the FINN partitioning workflow which creates multiple compute kernels that are floorplanned on a desired FPGA and a selected SLR (Super Logic Region) on it. We have used Xilinx's UDP stack and CMAC 100Gbps Ethernet IP to enable communication between the two FPGAs.

For this experiment, we need to create two FPGA bitstreams on MOC using the Xilinx Vitis and FINN tools. Then we transfer the bitstreams, driver files and input images to the respective hosts; load both bitstreams on the FPGAs; and configure the network stack by setting up IP addresses and ARP table entries to enable inter-FPGA communication. Finally, we initiate the accelerators by enabling data transfer from host to FPGA through the input DMA channels, and get the results through the output DMA channels. Table I shows

the performance for each accelerator. The color image size used was $224 \times 224 \times 3$, and FPGA clock frequency was 200MHz. Since both accelerators can be operated in parallel, the total throughput will be twice the value shown in the table. However, a monolithic design of the same accelerator can only be operated at about half of this frequency which results in a throughput of about 850 image/s. Therefore, partitioning a FINN graph into multiple compute kernels not only allows flexibility in the implementation, but also increases the throughput compared to a monolithic design. The communication latency between two FPGAs has no noticeable effect on the performance of the accelerator; i.e., the throughput of a 2-FPGA accelerator was nearly the same as that of a single-FPGA accelerator operating at the same frequency.

TABLE I
PERFORMANCE METRICS FOR EACH DISTRIBUTED ACCELERATOR.

FPGA Clock (MHz)	200
Batch Size	300
Runtime (ms)	182.8
Throughput (Images/s)	1641.1
Top-1 Inference Accuracy (%)	70

MobileNet was chosen as a proof of concept demonstrating that experimenters can implement a design split between two FPGAs with direct communications using the tools available in MOC. MobileNet is not memory bound nor does it require extra hardware resources; a single instance can fit on a single Alveo U280. We implement two instances and use the network connection to have the results of one labeling of an ImageNet dataset appear on a different host from the inputs, and the results of a second dataset appearing on a second FPGA. Even so, we use only a small fraction of the total available bandwidth. In this application, the input bandwidth is much higher than the output, as we read images on the input channel and only return inference labels on the output channel. The network connection is used to send activations between two FINN layers which uses about one third of the maximum memory bandwidth, i.e., about 85 MB/s. We are currently investigating the implementation of more complex neural networks such as ResNet-152 which cannot fully fit in a single Alveo U280, and require at least two such network attached devices.

IV. DISCUSSION

In this section, we discuss some of the implications that result from design choices we made when implementing the FPGAs in OCT.

First of all, as shown in Section III, each FPGA is connected with two full-duplex 100 Gbps Ethernet connections. This configuration will give users flexibility in the design of their distributed applications. For example, if an application's traffic is unidirectional, both interfaces can be used in parallel to increase the overall throughput between sending and receiving FPGAs, as was done with MobileNet. If traffic is more bidirectional, each of the two links can be used for traffic in one direction. This setup builds the basis for research in multi-path

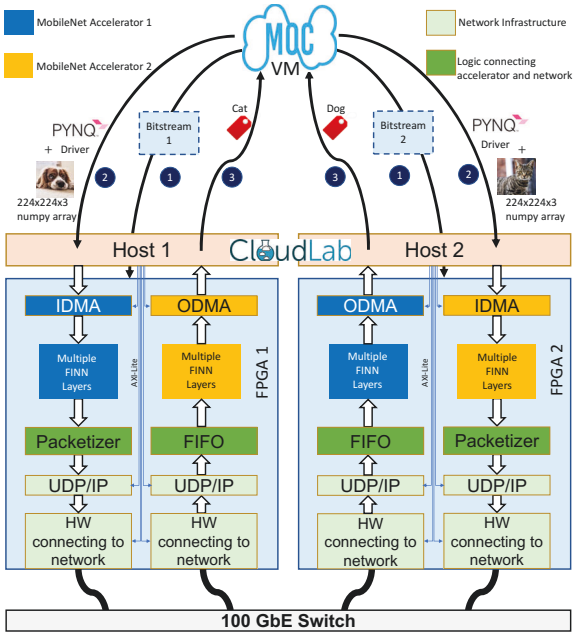

Fig. 7. MobileNet split across two network attached FPGAs.

protocols like MultiPath TCP [7] and MultiPath QUIC [9]. The fact that the combined throughput of both Ethernet interfaces (200 Gbps) is in the same ballpark as the maximum throughput of PCIe 3.0 bus (∼120Gbps uniderectional, 16-lane slot) of the host server also supports research and experimentation in the area of data center disaggregation. Note that in the current OCT setup, the FPGAs are all connected via the same Ethernet switch with separate collision domains providing high network efficiency. Finally, the OCT setup also allows the execution of two application in parallel, where each application can communicate via a dedicated Ethernet link.

The distributed MobileNet application (Fig. 7) does not saturate the network. We rather use it as a proof-of-concept to demonstrate that the FPGA cluster in OCT efficiently supports distribute applications on FPGAs and enables researchers to perform experiment in this area. Larger DNNs as, for example, residual networks with a depth of 152 layers will result in increased communication between FPGAs; however, the reason for splitting larger designs across multiple FPGAs is the need for more computational resources and memory. We are also investigating applications which require the bandwidth available through direct FPGA to FPGA communications; our throughput analysis in Section III shows that such applications can be supported. OCT will soon have sixteen FPGAs with required communication infrastructure to support scalability. This will enable implementation of complex designs at scale by floorplanning them across multiple FPGAs.

We recently started a new project that has the goal to develop a complete toolchain for enabling P4 and other types of advanced network-oriented programmability on FPGAs that will immediately benefit network researchers using the OCT and other testbeds that offer FPGAs as part of their

compute resources. The effort will provide FPGA building blocks integrated with P4 tools, and make use of available P4 tools that support FPGA development workflows. Testbeds or individual researchers will be able to take advantage of the building blocks and tools resulting from this effort. For example, researchers can use the FPGA resources in OCT and FABRIC [6] to perform wide-area, end-to-end experiments for distributed applications that run on FPGAs.

Finally, the FPGAs in OCT will enable researchers to implement and evaluate novel networking protocols like QUIC [17] or Data Center TCP (DCTCP) [2]. In addition, new protocol stacks that can operate on top of Ethernet can also be implemented and evaluated. Examples for such stacks are Named Data Networking (NDN) [16], which represents the new networking paradigm of Information Centric Networking [1].

V. Conclusion & Future Work

In this paper, we present the Open Cloud Testbed which has the goal to support research in novel cloud technologies. To achieve this goal, we built a testbed that can grow and shrink based on user demand by using resources from other clusters for certain periods. This paper focuses on the integration of FPGAs into OCT and the ability to perform experiments currently not possible in other testbeds. We use three different scenarios to demonstrate the types of systems research that can be enabled by the availability of networked FPGAs in bare metal servers. Results from our evaluation of these scenarios show that FPGAs can improve the performance of applications and offer new approaches to design and implementation. The results also show the importance of tuning TCP parameters for FPGA-to-FPGA and FPGA-to-Host communication. OCT is available to systems researchers and documentation is provided for getting started [14]. We encourage the community to make use of this resource to take networking to the next level.

Acknowledgment

This work was funded by National Science Foundation (NSF) grants CNS-1925464, CNS-1925504, CNS-1925658. All opinions and statements in the above publication are of the authors and do not represent NSF positions. We also thank the industry partners of the MOC and ORCI, especially Red Hat, Intel and Two Sigma for their financial, engineering, and in-kind support that have played a key role in the OCT, and Xilinx, Inc. for their generous donations to the OCT.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, et al. Data Center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference*, page 63–74, New York, NY, USA, 2010. ACM.

[3] G. Alonso, C. Binnig, I. Pandis, et al. Dpi: the data processing interface for modern networks. *CIDR 2019 Online Proceedings*, page 11, 2019.

[4] T. Alonso, L. Petrica, M. Ruiz, et al. Elastic-df: Scaling performance of dnn inference in fpga clouds through automatic partitioning. *ACM Transactions on Reconfigurable Technology and Systems*, 15(2):1–34, 2021.

[5] Amazon. Amazon EC2 F1 Instances. https://aws.amazon.com/ec2/instance-types/f1/. Accessed: 2021-12-29.

[6] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K. Wang, T. Lehman, and P. Ruth. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing*, 23(6):38–47, 2019.

[7] S. Barré, O. Bonaventure, C. Raiciu, and M. Handley. Experimenting with multipath TCP. *SIGCOMM Comput. Commun. Rev.*, 41(4), Aug. 2010.

[8] M. Blott, T. B. Preußer, N. Fraser, et al. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):16, 2018.

[9] Q. De Coninck and O. Bonaventure. Multipath quic: Design and evaluation. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, pages 160–166, New York, NY, USA, 2017. ACM.

[10] D. Duplyakin, R. Ricci, A. Maricq, et al. The Design and Operation of CloudLab. In *USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1–14, July 2019.

[11] ETH Zurich. Vitis with 100 Gbps TCP/IP Network Stack. https://github.com/fpgasystems/Vitis_with_100Gbps_TCP-IP. Accessed: 2021-12-29.

[12] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, et al. Azure accelerated networking: Smartnics in the public cloud. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 51–66, 2018.

[13] P. Haghi, A. Guo, Q. Xiong, R. Patel, C. Yang, T. Geng, J. T. Broaddus, R. Marshall, A. Skjellum, and M. C. Herbordt. FPGAs in the Network and Novel Communicator Support Accelerate MPI Collectives. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–10, Waltham, MA, USA, Sept. 2020. IEEE.

[14] S. Handagala. OCT FPGA Tutorials. https://github.com/OCT-FPGA. Accessed: 2021-12-29.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, page 1–12. ACM, 2009.

[17] A. Langley, A. Riddoch, A. Wilk, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 183–196. ACM, 2017.

[18] M. Leeser, S. Handagala, and M. Zink. FPGAs in the Cloud. *IEEE Computing in Science Engineering*, 23(6):72–76, 2021.

[19] S. Mosanu. Aes implementations. https://github.com/hplp/AES_implementations, 2019.

[20] D. Sidler, Z. Istvan, and G. Alonso. Low-Latency TCP/IP Stack for Data Center Applications. In *Field Programmable Logic (FPL)*, 2016.

[21] N. Tarafdar, N. Eskandari, T. Lin, and P. Chow. Designing for fpgas in the cloud. *IEEE Design & Test*, 35(1):23–29, 2017.

[22] A. Turk, R. S. Gudimetla, E. U. Kaynar, J. Hennessey, S. Tikale, P. Desnoyers, and O. Krieger. An experiment on bare-metal bigdata provisioning. In *8th USENIX Workshop on Hot Topics in Cloud Computing (Hot-Cloud 16), Denver, CO*, 2016.

[23] FINN Examples. https://github.com/Xilinx/finn-examples#example-neural-network-accelerators.

[24] Xilinx. Xilinx adaptive compute cluster (XACC) program. https://www.xilinx.com/support/university/XUP-XACC.html.

[25] Xilinx. XUP Vitis Network Example (VNx). https://github.com/Xilinx/xup_vitis_network_example. Accessed: 2021-12-29.

[26] J. Zhang, Y. Xiong, N. Xu, R. Shu, B. Li, P. Cheng, G. Chen, and T. Moscibroda. The Feniks FPGA Operating System for Cloud Computing. In *ACM APSys*. ACM, September 2017.

[27] M. Zink, D. Irwin, E. Cecchet, H. Saplakoglu, O. Krieger, M. Herbordt, M. Daitzman, P. Desnoyers, M. Leeser, and S. Handagala. The open cloud testbed (oct): A platform for research into new cloud technologies. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pages 140–147, 2021.