2022

FPGAs in the datacenter: enhancing performance, usability, security, and privacy

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

FPGAS IN THE DATACENTER: ENHANCING PERFORMANCE, USABILITY, SECURITY, AND PRIVACY

by

RUSHI PATEL

B.S., Carnegie Mellon University, 2010
M.S., Boston University, 2017

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2022

Approved by

First Reader

Martin C. Herbordt, PhD
Professor of Electrical and Computer Engineering

Second Reader

Orran Krieger, PhD
Professor of Electrical and Computer Engineering

Third Reader

Wenchao Li, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Systems Engineering

Fourth Reader

Mayank Varia, PhD
Associate Professor of Computing and Data Sciences

Acknowledgments

First, I would like to express my appreciation and gratitude to my advisor, Prof. Martin Herbordt, for his continuous support, trust, and guidance throughout my PhD. I also want to give special gratitude to Prof. Mayank Varia, who was instrumental in my progress toward my PhD with limitless help and guidance on my research. I would like to thank Prof. Orran Krieger, who has been in my committee through out my entire PhD, starting with my RPC entrance exam and finally my thesis defense. I appreciate Prof. Wenchao Li agreeing to join my committee and providing a fresh perspective and insightful comments.

I want to thank Jeff Hughes, Nabil Schear, Charles Munson, and all the folks at MIT Lincoln Laboratory for the welcoming, wonderful, and enlightening experience. I would not have found such an interest in my research without my internships with group 53. I also want to thank Venkata Krishnan, Tina Zou, and the team at Intel for all the help and time they spent answering my questions, providing me with guidance, and the invaluable experience.

I give a special acknowledgement to Pierre-François Wolfe who worked closely with me during and after his master's program. I am happy our too-brief graduate program experience turned into a lasting friendship. I want to share my gratitude to Robert Munafo for all the feedback and overall being a friend. To all the other members of the CAAD lab, current and graduated, a sincere thank you; and all my Boston University friends who have always been friendly and supportive for everything.

A very special thank you to Leila Delshadtehrani for all the encouragement, support, and overall time spent together from day one and looking forward to many more years together. Last but not least, I am grateful to my parents and family for their continuous encouragement and support of my success in both life and future goals.

–Rushi Patel

FPGAS IN THE DATACENTER: ENHANCING PERFORMANCE, USABILITY, SECURITY, AND PRIVACY

RUSHI PATEL

Boston University, College of Engineering, 2022

Major Professor: Martin C. Herbordt, PhD
Professor of Electrical and Computer Engineering

ABSTRACT

One of the great advances in computation over the last 20 years is the availability, connectivity, and growth of datacenters. These advances in the datacenter have motivated research, development, and adoption of new techniques to improve the overall performance of datacenter activities. Such techniques include virtualization, software defined networking (SDN), distributed computing, platform-as-a-service (PaaS), serverless computing, and many more. Continued improvements and support for all these datacenter applications has required a re-imagining of the current connected environment into one that is more data focused, where computation is everywhere, and high performance networks connect all resources together. A key part of this evolution is making network fabrics more intelligent by substituting traditional network interface cards (NICs) with SmartNICs; these create opportunities to move computation away from compute nodes and into the network. Field programmable gate arrays (FPGAs) play a significant role in realizing these new opportunities in the network fabric as they have both compute and communication hardware. In this thesis, we investigate performance and usability of network FPGAs and demonstrate

their utility with case studies in security and privacy for tenants in the datacenter.

We propose an approach to achieve a new level of security and performance through the use of reconfigurable hardware available with network attached FPGA SmartNICs. We first show how FPGAs directly connected to high-bandwidth network resources can reduce the network load through function offload onto FPGAs. We then demonstrate the capabilities of FPGA SmartNICs to perform in-line packet acceleration functions as well as application acceleration without host processing. Next, we improve the security of the datacenter through network isolation techniques and secure data processing between tenants co-located in the same datacenter. To this end, we accelerate a state-of-the-art cryptographic algorithm, Secret Sharing Multi-Party Computation (MPC), using FPGAs. Our FPGA accelerated Secret Sharing MPC uses at least a 10× less computing resources compared to the original design using CPUs.

To demonstrate the real-world advantages of leveraging network-attached FPGA SmartNICs, we present three unique applications of these FPGAs specifically for security. First, we improve the performance of datacenter nodes on CPUs by offloading common packet functions such as remote memory access and packet fragmentation. Second, towards the goal of providing private cloud enclaves, we propose a method of securing tenant private networks with FPGA SmartNICs. Finally, we leverage FPGA SmartNICs to efficiently perform joint confidential data processing between cooperating organizations.

Contents

List of Tables

List of Figures

List of Abbreviations

2PC	Two Party Computation
3PC	Three Party Computation
AES	Advanced Encryption Standard
AI	Arithmetic Intensity
API	Application Programming Interface
ARM	Advanced RISC Machines
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instruction Set Processor
AWS	Amazon Web Services
AXI	**A**dvanced e**X**tensible **I**nterface
BitW	Bump in the Wire
BU	Boston University
CAAD	Computer Architecture and Automated Design (Lab)
CCPA	California Consumer Privacy Act
CLaaS	**CL**assified as a **S**ervice
COPA	**CO**nfigurable Network **P**rotocol **A**ccelerator
CPU	Central Processing Unit
CSfC	Commercial Solutions for Classified
DMA	Direct Memory Access
DoS	Denial of Service
DP	Differential Privacy
DRAM	Dynamic RAM
DSP	Digital Signal Processor
EC2	(Amazon) Elastic Compute Cloud
FaaS	Function as a Service
FedRAMP	Federal Risk and Authorization Management Program
FL	Federated Learning
FPGA	Field-Programmable Gate Array
FSS	Firmware Secure System
GC	Garbled Circuit
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HDD	Hard Disk Drive
HDL	Hardware Description Language
HIL	Hardware Isolation Layer

HIPAA	Health Insurance Portability and Accountability Act
HLS	High-Level Synthesis
HPC	High-Performance Computing
IaaS	Infrastructure as a Service
IoT	Internet of Things
IP	Intellectual Property
LAN	Local Area Network
MAC	**M**ultiply **AC**cumulate
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MM	Matrix Multiply
MOP	**M**PC **OP**eration
MPC	Multi-Party Computation
MPCaaS	MPC as a Service
MPI	Message Passing Interface
MTU	Maximum Transmission Unit
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OpenCL	Open Computing Language
OS	Operating System
PaaS	Platform as a Service
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PHI	Protected Health Information
PRF	Pseudo-Random Function
PUF	Physical Unclonable Function
RAM	Random-Access Memory
RDMA	Remote DMA
RISC	Reduced Instruction Set Computer
RNG	Random Number Generator
R&D	Research and Development
SaaS	Software as a Service
SDK	Software Development Kit
SDN	Software Defined Networking
SFE	Secure Function Evaluation
SHA	Secure Hash Algorithms
SoC	System on Chip
TCP/IP	Transmission Control Protocol / Internet Protocol
VLAN	Virtual LAN
VM	Virtual Machine

Chapter 1

Introduction

1.1 Overview

One of the great advances in computation over the last 20 years is the availability, connectivity, and growth of datacenters. These advancements in the datacenter have led to the adoption of new techniques to improve the overall performance of datacenter activities. Such techniques include virtualization, software defined networking (SDN), distributed computing, platform-as-a-service (PaaS), serverless computing, and many more. Continued improvements and support for all these datacenter applications require a re-imagining of the currently connected environment. The future is evolving toward a more data focused environment where computation is everywhere and high performance networks connect all resources together. A key part of this evolution is making network fabric more intelligent by substituting traditional network interface cards (NICs) with SmartNICs. The inclusion of SmartNICs provides additional capabilities in datacenters, e.g., edge computing, in-network computing, and network attached storage. These capabilities create opportunities to move computation away from compute nodes and to the network. Field programmable gate arrays (FPGAs) play a significant role in realizing these opportunities as they have both compute and communication hardware. In this thesis, we propose leveraging the capabilities of FPGAs to enable methods of improving security and privacy for tenants in the datacenter.

Over the last decade, maintaining security and privacy has become one of the most

important challenges in computing systems. An increasing number of organizations are adopting more advanced security and privacy techniques while using public datacenters. An emerging approach to address security concerns in public datacenters is leveraging private cloud enclaves. Such enclaves offer tenants security while using public resources through methods of hardware isolation and network protection. Inside an enclave, organization can use custom encryption algorithms to achieve higher security guarantees. Adding network attached FPGAs to private cloud enclaves enables tenants to upgrade the encryption algorithms as security threats evolve. Although private cloud enclaves provide privacy through isolation techniques, they are costly and restricted to a small subset of tenants. A potential solution for datacenter privacy could be through the use of cryptographic algorithms such as Multi-Party Computation between public cloud tenants. Unfortunately, such crytographic algorithms are computation and/or communication intensive and running these algorithms on CPU-centric datacenters is not viable. As FPGAs are effective at accelerating both computation and communication heavy applications, they provide a perfect framework for running cryptographic algorithms in the datacenter. In this thesis, we propose to leverage network connected FPGAs to achieve security and privacy in the datacenter.

1.2 Datacenters and the Move to Smart Network Fabrics

In recent years, one of the biggest technological advancements comes from the growth of public datacenters. The distributed computing environments enable us to continuously surpass hardware limitations. These days, organizations have access to hundreds of hardware resources without the need to invest large amounts of time and money in building private datacenters. Due to the increasing demand for access to datacenter resources, cloud providers strive to find new ways of increasing the overall datacenter performance. Unfortunately, the predicted performance improvement by

Moore's Law does not hold true anymore, which forces cloud providers to invest in alternative methods to upgrade datacenters. Application acceleration utilizing graphics processing units (GPU), field programmable gate arrays (FPGA), and application-specific instruction set processor (ASIP) provides new avenues to improve the overall performance of the datacenter.

GPUs help provide unique computation performance for highly parallel applications and enable achieving higher performance compared to multi-core CPUs. Software libraries such as CUDA and OpenCL enable software developers to utilize GPU resources without an exact knowledge of the underlying architecture (Du et al., 2012). GPUs continue to be an important hardware accelerator for many high-performance computing applications (Landaverde and Herbordt, 2014; Sanaullah et al., 2016c; Sukhwani and Herbordt, 2014; Wu et al., 2021a).

Recently, FPGA as co-processor accelerators are integrated into datacenters to provide dynamic reconfigurable hardware for accelerating big data processing applications. FPGA is a versatile hardware resource using an array of programmable logic blocks and interconnects that allow the reconfiguration of hardware based on the programmer's requirements. Initially, FPGAs were widely used as communication processors and aided in the telecommunication and networking industry. Now FPGAs are utilized as co-processors in high performance applications which benefit from hardware reconfiguration. Such utilization have shown major improvements to a range of HPC applications (Herbordt and VanCourt, 2005; VanCourt and Herbordt, 2005; VanCourt and Herbordt, 2006a; VanCourt and Herbordt, 2006b; Gu et al., 2006; Sanaullah et al., 2016a; Sheng et al., 2014; Yang et al., 2019b; Yang et al., 2019a).

The public cloud has recently started migrating into a more data-centric system where all hardware is interconnected and shared using high speed networks. The in-

troduction of data processing units (DPUs) and SmartNICs (Zilberman et al., 2014; NVIDIA, 2022b; NVIDIA, 2022c) aim to enable connectivity between tenants with all available public cloud resources. These new connectivity options improve upon both the computation bottlenecks of tenant applications and infrastructure requirements of public cloud vendors. Prior to dedicated hardware options, management of cloud resources was done using the compute resources of each individual node. Nodes would dedicate entire cores for management tasks such as network provisioning, hypervisors, and software security reducing the overall efficiency of cloud nodes. Each additional CPU core used by the cloud provider for management tasks reduces the overall performance of a given node for tenant applications.

DPUs are a combination of high-speed networking connectivity, a dedicated high-speed network processor, and a low power onboard processor which runs an embedded real-time operating system (RTOS) (Fungible, 2022; NVIDIA, 2022b). Through their unique host-independent capabilities, DPUs enable cloud providers an effective method to offload management features such as virtual machine provisioning and isolation, network resource sharing, and remote hardware access. As DPUs use low power processing cores, acceleration options are limited to low performance data manipulation.

FPGA SmartNICs are an alternative to DPU resources which provide similar functionality in management and acceleration options without the need for entire an operating system and processing cores to perform software defined functions. FPGA SmartNICs are a compelling option to provide hardware only solutions for in-line packet processing or offload acceleration. Headless operation in FPGA SmartNICs delivers similar advantages to DPUs in terms of management offload features. Utilization of network attached FPGAs in commercial clouds is limited to in-house operation and is not available for public cloud tenants.

While in early days accelerator options were limited to dedicated cloud nodes and were only offered to customers with special requests and application demands, nowadays the public cloud aims to support various organizations and their specific application needs. However, utilizing shared public resources in datacenters is not practical for organizations which require higher security guarantees. In this thesis, we demonstrate that the availability of FPGA SmartNICs to both cloud providers and tenants offers unique features for security related applications.

1.3 Security in the Datacenter

Growing data requirements give rise to the importance of data privacy for individuals and organizations. Data privacy is an important issue as organizations collect and utilize increasing amounts of data every day with some having little concern over the data holder's privacy.

Data leaks and ransomware attacks occur more regularly to companies with little to no security mechanisms in place. These data leaks put the privacy and security of individuals at risk and cause very substantial problems for individuals with no control over the data collected by organizations. Due to this high potential damage, control over private and classified data continues to be important for all data holders, both individual and organizations.

Organizations with security and privacy as a top priority traditionally avoid the public cloud as they lack control over the environment and require a significant amount of trust with the cloud provider and their security practices. These organizations rely on private on-site datacenters as one of the only options to utilize advancements in distributed computing for confidential data processing. Some companies may want to share data with like-minded organizations as a global data set provides advantages for many big data applications. Those organizations with an

interest in data privacy do not have a simple solution to combine data with another entity without the risk of leakage or loss in confidentiality. Having a secure and easy method to sharing confidential data can open up a number of possibilities to any organization limited in classified data processing.

There is a growing need to improve the security and privacy of the public cloud to help enable future avenues for confidential data applications and individual data protection. Addressing the security of cloud tenants and improving methods of secure joint computation lead to improved security mechanisms while using the public cloud. Improved security in the public cloud reduces the chance of future data leaks for all tenants. Overall improved security is not just limited to those organizations with strict confidential data regulations, but can have a positive effect for all companies and individuals using public resources.

The introduction of regulations for consumer data such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA) are a start to help individuals maintain control of their data. The purpose of these regulations is to enable protection for individual data owners over their own data. These regulations aim to provide individual data privacy, but should instead be used by organizations as motivation for improved security of private data. This is similar to organizations such as government, healthcare, and finance which are already under strict regulations when it comes to handling classified data.

1.3.1 Example Scenarios

The Health Insurance Portability and Accountability Act (HIPAA) was established in 1996 as a federal law to protect the sensitive data of individual health records from being disclosed without the consent of data holders or those with explicit control of individual data. This law means healthcare facilities must always value the data of individual patients as a top priority and must maintain full security and privacy

over data collected and utilized for further research or processing. This control on healthcare organizations inhibits the continued growth of research as they would benefit from an increased amount of raw data access.

Government organizations are another example with special classified data requirements. Government security requirements must meet standards classified into three categories, confidentiality, isolation, and assurance. Each category represents security thresholds based on need and government use cases. Confidentiality is the act of providing privacy on data-at-rest and data-in-transit from any outside malicious actors. Isolation is the method of protecting government resources from attacks either remote or physical. Assurance is the requirement to enable access to each resource without any interference or downtime. This is important as critical systems must be available during any catastrophic event scenarios. The combination of each security requirement can add up to a very strict security guarantee. This is why traditionally the government is unable to use public resources for classified data use cases.

1.3.2 Public Cloud Security

An alternative to in-house private datacenters, public clouds offer special accommodations for private data organizations in the form of separated hardware silos. These "walled garden" offerings add security through separation from public tenants and improved security guarantees for data storage and communication. They are typically removed from the public hardware datacenter and placed into physically enclosed environments with limited access by authorized individuals. Due to the increased security protections, hardware in these private accommodations typically undergo slow update cycles and are often behind in hardware generation. Public cloud vendors are aware of the shortcomings of these private datacenter silos and have worked with the US Federal Government to offer an security-conscious alternative known as FedRAMP (VanRoekel, 2011). However, these offerings still require increased cost and

maintenance which limits the overall adoption by all classified data organizations.

Researchers are investigating approaches to create dynamic private enclaves with public cloud bare metal nodes (Mosayyebzadeh et al., 2019). The goal is to reduce the trust required using a cloud provider and allow for private enclaves of cloud resources only accessible by private tenants. An organization can request for a private enclave which then creates a private network and resource pool carved out from the public resource allocations. The organization can then act like a cloud provider for private entities and distribute their enclave resources with no authorization and connection with the public cloud provider. Creating private cloud enclaves requires vendors to hand over full control of public cloud resources and remove any additional monitoring tied to each instance. This can cause disruption and additional headache for cloud vendors, thus has not be developed further outside a research space.

1.3.3 Privacy Preserving Computation

A current field of study is the usage of privacy preserving computation as a method of secure computation over private data. Confidential data is first encrypted as a means to obfuscate it from any public attackers. Computation is performed on the encrypted versions of data by any public facing system. Upon completion of all computation, data is decrypted and the final value is reconstructed into the plain-text version.

The main goal of privacy preserving computation is the ability to share private data and compute an agreed upon computation without the fear of data leakage during the computation process (Kissner and Song, 2005). Computation can be performed using public resources while maintaining privacy as all calculations are performed on encrypted data. This allows companies with sensitive data to participate in using public clouds or allow third party entities to perform computation behalf data holders without knowledge of the private input data.

All computation can be converted into privacy preserving computation but may

require a few changes for added security and performance. Such changes include rethinking how sensitive data is consumed and re-engineering of the applications. An example of re-engineering of applications can be seen in the rise of Federated Learning (FL) (McMahan et al., 2016). Federated Learning allows many participating entities to aid in the training of machine learning models utilizing their private data. One method of Federated Learning is to perform training updates locally at each data holder and then aggregate each updated model using a trusted third party. A real world example is the auto-completion model for google keyboard, which utilizes information from individual android users without sharing each individuals most commonly used words (Bonawitz et al., 2017).

Secure joint computation can be utilized for all arithmetic and logic applications as the base implementation replaces simple functions such as addition and multiplication with secure versions. This can introduce a large performance loss compared to operations with plain-text data, but would enable secure processing at the lowest level.

1.3.4 Advances in Public Cloud Infrastructure

Advances in the datacenter through the use of application specific acceleration such as GPU, FPGA, and ASIC hardware aim to improve upon hardware bottlenecks. One example is the inclusion of GPU hardware as a prolific improvement to highly parallel applications such as machine learning on efficient parallel hardware. The introduction of FPGAs as co-processor accelerators are an attempt to introduce dynamic reconfigurable hardware to address applications which may not fit well on GPU resources. The FPGA is a versatile hardware resource which greatly benefit applications that have high communication demands.

FPGA hardware are used as network processing accelerators and network attached headless nodes to provide bare metal functionality with no host system requirements.

Commercial utilization of network attached FPGAs in the datacenter is limited to in-house acceleration and do not provide methods for public tenants to use this hardware environment. Datacenters continue to adopt FPGAs into many different roles as they offer customized improvements to common datacenter requirements. These include co-processors (Amazon, 2022a), network attached FPGA SmartNICs (Caulfield et al., 2016; Chung et al., 2018), dedicated accelerator network (George et al., 2016), and storage devices (Sakalley, 2017).

1.4 Thesis Statement and Contributions

The growing need to maintain security over private data was once a restriction on a select few organizations, however continues to become a growing concern to more companies and individuals alike. There is no single best method to improve upon current security guarantees while also enabling future advancements in applications and technology. This thesis aims to provide a multi-pronged approach to utilizing hardware available in the public cloud to both enforce privacy while also limiting the impact on performance when adding security minded approaches to existing methods. We utilize the customization and reconfigurability of commercial FPGA hardware to enable security features for all tenants of a public cloud and enable privacy aware methods of joint computation. We present our thesis statement as follows:

Network connected FPGAs can improve public cloud security and privacy for confidential data holders by enabling high performance secret sharing MPC and enhancing network isolation techniques.

The main contributions of this PhD research are as follows:

- **Demonstrate network offload improvements for security and performance leveraging network connected FPGAs.** Improving the performance of applications in the datacenter require a re-imagining of distributed

systems and hardware connectivity. Currently, distributed applications utilize many core processors and direct memory across multiple network connected nodes. Control and management of each network connected node is managed with the local CPU resources available. Support and cost of CPU only compute nodes increases every year as improvements to performance continue to decline. An alternative to CPU only systems is required to improve upon the performance of datacenter activities. We demonstrate hardware accelerated offload options using FPGA SmartNICs for both applications and cloud management. Offload onto the network fabric alleviate CPU resources and allow focus to remain on tenant applications. The demonstrated offload options help enable a future data-centric model where computation is spread out across the datacenter.

- **Utilize FPGA resources for improved network security with little to no dependency on the host system.** Government regulations utilize a two layer protection approach when communicating over a public network. They maintain that two unique methods of protection must be placed between restricted devices when utilize public channels. This approach can be useful for many organizations wanting to secure communication between public cloud nodes in a datacenter. We propose the use of network attached FPGAs to create a secure hardware encryption layer between each individual public cloud node. Maintaining a secure independent network device unassisted by a host node helps to provide a physical separation from a potentially compromised system. Our goal is to enable a customizable and cost-efficient method of providing an additional layer of security for organizations participating in the public cloud while minimizing the trust required of cloud providers.

- **Enable hardware accelerated secure joint computing in the datacenter**

with multi-party computation. We propose that secure joint computation is best done in a datacenter setting through the use of secret sharing multi-party computation. The datacenter provides the ideal environment to best use privacy preserving algorithms for joint computation. Low latency networks and high throughput computation can help achieve performance results which enable real world use cases for multi-party computation. We employ hardware accelerated multi-party computation with a network of FPGAs to remove the current bottlenecks that are faced with traditional MPC operations in both computation and communication. This enables secret sharing MPC to be a viable solution for secure joint computation between party members with restrictions on confidential data. Through the combination of performance improvements to both computation and communication, we demonstrate that utilizing FPGAs outperforms the same algorithms running on a high core CPU and NIC.

1.5 Organization

The remainder of this dissertation is organized a follows. We start with background and related work in chapter 2, where we give background of public cloud datacenters, the need for secure private cloud enclaves, privacy preserving computation and utilization of FPGAs for enhanced computing. After that, we discuss how the datacenter is migrating away from compute-centric and into a data-centric model and how network communication is becoming the essential backbone of the datacenter in chapter 3. In the next few chapters, we cover three pieces of our work, including network function offload with FPGA SmartNICs in chapter 4, secure private enclaves for classified usage in chapter 5, and secure joint computation using hardware accelerated MPC in chapter 6. Finally, we present a conclusion and an overview of future work in chapter 7.

Chapter 2

Background and Related Work

In this chapter, we give background on the public cloud and available security options for tenants and private organizations. In section 2.1 we first look into current public cloud offerings and how they have changed over the past few years. We next introduce the currently available options for private cloud enclaves and their limitations in section 2.2. Finally, we compare the different methods for privacy preserving computation in section 2.3.

2.1 Public Cloud Datacenter

Growth of the public cloud datacenter continues to accelerate every year with the inclusion of higher core count processors and faster networks between multiple nodes. The major cloud providers spend enormous amounts of money providing incremental improvements to the cloud ecosystem targeting high performance computing (HPC) applications such as machine learning and artificial intelligence. However, the scale of improvements continues to decline as predicted performance improvements by Moore's Law do not hold true anymore. Companies begin to look into alternative methods of improving the datacenter environment through application specific hardware offload. Vendors such as Amazon and Microsoft show they are willing to invest in unique hardware offerings to enable new avenues of technology growth. Instances containing GPUs and FPGAs help to provide tenants a chance to develop with cutting edge hardware new technology which exceeds the current compute bound applications

in the HPC datacenter. In addition, these cloud vendors also utilize these unique hardware platforms to improve upon the operations running in the datacenter. One example we will discuss further is Microsoft and their inclusion of FPGA hardware for network function acceleration.

Early days of public clouds, growth was based directly on the number of offerings available through high core count CPUs and large memory nodes. Investment was made to increase the size of both number of cores available and total amount of memory accessible for each tenant virtual machine (Amazon, 2022b). Amazon offers today the choice of high core count systems with 192 core vCPU and high memory systems with up to 24 TiB of accessible memory. As the trend of high core offerings become increasingly more difficult to improve upon, Amazon began to offer GPU instances back in 2016. These P2 instances enabled tenants to run GPU workloads on dedicated virtual nodes with up to 16 GPU cores available per instance. Tenants were quick to adopt these instances for HPC applications such as machine learning and deep learning. The increased compute resources provide a large improvement in parallel core count compared against a CPU only system. Following the success of GPU instances, Amazon then began introducing FPGAs similarly to how GPUs are utilized as co-processor accelerators (Amazon, 2016). The first F1 instances also introduced in 2016 enabled use of up to 16 FPGAs in each node running Xilinx Virtex UltraScale+ hardware. AWS provides either an HDL programming environment with all the provided interfaces between the host system or OpenCL HLS with their own unique board support package (BSP) encapsulating the Virtex FPGA.

Utilizing all the available compute resources, the growth of large-scale distributed applications continues to take over the datacenter. Here multiple nodes are utilized together to deliver a pool of resources to perform distributed processing. Common programming models such as MapReduce (Dean and Ghemawat, 2008) utilize many

Table 2.1: AWS Instances

Instance Size	vCPU	Memory (GiB)	Network BW (Gbps)	GPU	FPGA
c6a.large	2	4	12.5	0	0
c6a.xlarge	4	8	12.5	0	0
c6a.48xlarge	192	384	50	0	0
u-24tb1.metal	448	24576	100	0	0
g5.xlarge	4	16	10	1	0
g5.48xlarge	192	768	100	8	0
p2.16xlarge	64	732	20	16	0
f1.2xlarge	8	122	10	0	1
f1.16xlarge	64	976	25	0	8

nodes and many processes to perform distributed tasks such as database lookup, key-value store, and scatter-gather operations. The goal of distributed models is to provide mechanisms to scale up applications to utilize as many resources necessary to reduce the overall latency of highly parallel tasks. Orchestrating the communication and data access of the distributed nodes requires a large amount of network and IO for communication and data access.

2.2 Secure Public Cloud Enclaves

Government and private data organizations utilize public clouds in a unique manner from traditional tenants. They typically utilize physically isolated private instances of the datacenter only accessible by one organization (Amazon, 2022c; Microsoft, 2022). The physically isolated system limits any unauthorized physical access to help mitigate the potential attack surface requiring access to the underlying hardware. This could include man-in-the-middle attacks or side channels associated to power and hardware snooping (Agrawal et al., 2003; Zhao and Suh, 2018; Lin et al., 2009). As shown in Figure 2·1, public clouds have various attack vectors which reduce the security guarantees required for the confidential data clients such as government.

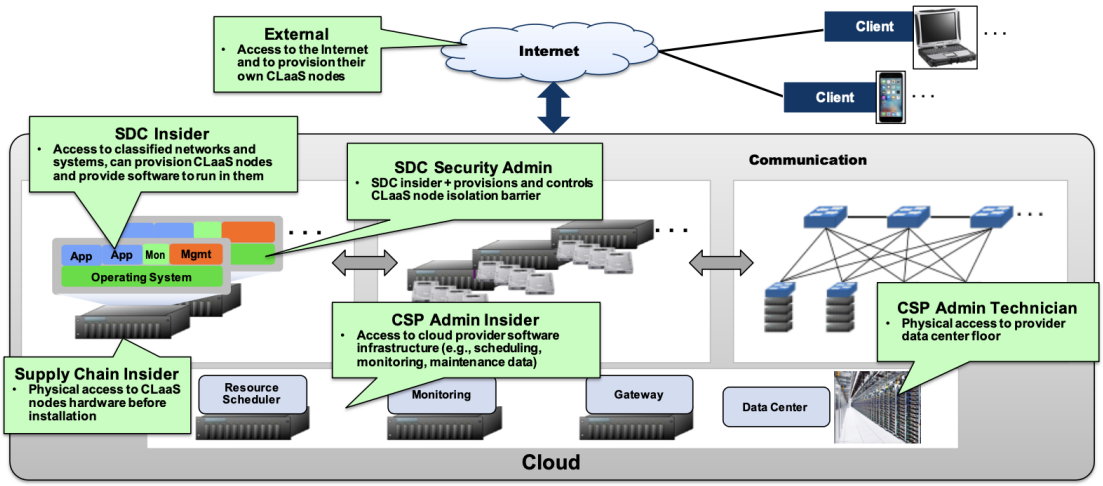


Figure 2·1: Public cloud threat actors and their implicit access

These attack vectors could be in the form of the manufacturing process for each node (Tehranipoor and Koushanfar, 2010), unauthorized software access by cloud provider administration, the maintenance crew with remote and physical access to the datacenter hardware, and other tenants utilizing the same shared datacenter resources.

Outside of the datacenter, government entities follow a strict guideline when accessing private networks both intranet and through the external web (National Security Agency, 2022). Through regulation and policy descriptions, requirements include utilizing at least two layers of encryption from two different manufacturing sources. This includes usage of Type 1 encryption and commercial-off-the-shelf encryption devices (Smid and Branstad, 1988). These regulations are not limited to just government usage but can also be utilized for organizations accessing public cloud instances.

As an alternative, there is a private cloud marketplace option known as FedRAMP (VanRoekel, 2011) which allows for public cloud vendors to provide their private secure instances as potential options for use by these security conscience tenants.

These options usually attempt to secure public hardware in a cost efficient manner. AWS for example offers their C2S private cloud (AWS Public Sector Blog Team, 2017) which separates the datacenter away from the public cloud offerings. Microsoft Azure attempts the same thing as Amazon through isolated nodes and networks. This however results in limited upgrades and refresh cycles, as the hardware is conforming to the FedRAMP requirements and is less often updated.

2.3 Privacy Preserving Computation

Privacy preserving computation is a method of utilizing public cloud resources for private data computation. Public cloud resources can be utilized for computation on private data through the use of crytographically secure protocols which enable a set of unique arithmetic functions on encrypted data. There are two major classification of privacy preserving computation, Homomorphic Encryption and Multi-Party Computation.

2.3.1 Homomorphic Encryption

Homomorphic Encryption is considered the 'holy grail' of secure data computation as these methods would provide computation on encrypted data by any untrusted entity (Will and Ko, 2015). Utilizing homomorphic encryption protocols, private data holders are able to utilize public cloud resources and algorithms to perform operations on their private data without the risk of data leakage. A prime candidate for homomorphic encryption is performing statistical analysis on the private financial data of individuals in a corporation. For example, if accountant *Alice* stores all client information on a public server in encrypted formats. She wants to take a statistical average of all individuals with a specific common account. Without homomorphic encryption, she would need to decrypt all account holder data to perform her statistics operation, however with homomorphic encryption she can utilized the data in

encrypted format and only decrypt the final value without ever knowing the initial input private data.

Currently there are three types of homomorphic encryption:

- Partially Homomorphic Encryption (PHE) - Only a single type of arithmetic operation is available and can be performed an unlimited number of times, e.g., addition or multiplication.
- Somewhat Homomorphic Encryption (SHE) - All mathematical operations are available for use with only a limited number of operations done in sequence. This is due to values losing accuracy over time and encrypted forms must be reconstructed to continue arithmetic operations.
- Fully Homomorphic Encryption (FHE) - All operations can be performed an unlimited number of times.

Homomorphic encryption sounds like a potential solution to our problem, but in reality this form of computation still remains in development as it suffers from high resource overhead and complexity (Cheon et al., 2017). Alternative types such as PHE and SHE first started as stepping stones to a FHE algorithm, but continue to be used for higher performance and better accuracy compared to complex methods of FHE. Some shortcomings to fully homomorphic encryption include higher data requirements and much larger computation resource utilization to perform simple arithmetic operations. FHE requires much larger data sizes for cipher-test representations compared to their clear-text versions as a set of input values are converted into a polynomial expression. Operations are then performed on the cipher-text polynomials where noise is introduced into the polynomial. After a limit of sequential operations is reached, a bootstrapping mechanism helps to recreate a new polynomial expression that eliminates much of the noise introduced.

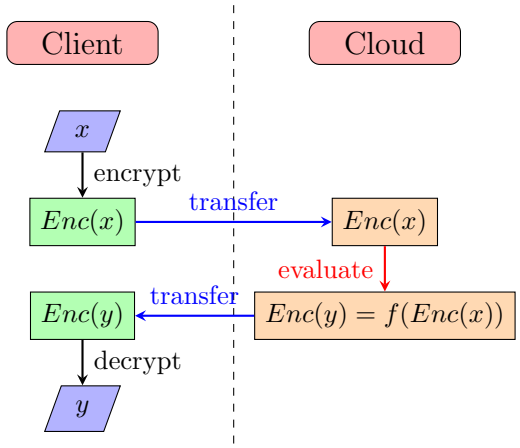


Figure 2·2: Fully Homomorphic Encryption

As imagined, the base operations on Fully Homomorphic Encryption take hundreds of cycles longer than plain-text operations and thus require much more time and computation resources.

2.3.2 Secure Joint Computation - Multi-Party Computation

Secure joint computation of data is the method of performing computation with other like-minded individuals with the goal of obtaining a collaborative result between members. All this is done while maintaining a level of security and trust between all participants in the process. This enables all protected data to be utilized without the risk of data leakage. Secure Multi-Party Computation (MPC) is one example of data computation between a few party members which maintains a high level of confidentiality since during computation trusted party members never learn more than what is required to continue computation. MPC has been an active area of research for the last 40 years (Evans et al., 2018; Yao, 1982; Yao, 1986; Shamir, 1979), and it has been deployed to protect data in the healthcare (Archer et al., 2018; Giannopoulos and Mouris, 2018), education (Bogdanov et al., 2016; Feigenbaum et al., 2004), finance (Bogetoft et al., 2009; Damgård et al., 2017; Abidin et al., 2016), and

technology (Bonawitz et al., 2017; Ion et al., 2019). Since then MPC has continued to evolve to provide highly efficient protocols while continuing to maintain a minimum level of trust and minimum chances for data leakage (Araki et al., 2016b; Araki et al., 2017; Demmler et al., 2015; Mohassel and Rindal, 2018; Wagh et al., 2020; Byali et al., 2020; Dalskov et al., 2021; Chaudhari et al., 2020).

Garbled Circuits

The first of these being Andrew Yao's approach formally known as Garbled Circuits. Originally from 1986 (Yao, 1982) was used in a thought experiment to solve a simple problem, without disclosing any information does member A have more or less money than member B. Following this, he came up with an approach in 1986 that formally fulfils the thought experiment. This approach uses a sender and receiver protocol where one member generates a cryptographically secure algorithm for the other to evaluate. During circuit generation, the sender creates a circuit for evaluation by generating labels for each gate input and assigns the outputs as encrypted versions based on theoretical inputs selected. For each gate in the circuit is represented through a table of encrypted entries which are only decrypted with the correct labels versions of outputs based on the inputs to the gate, as shown in Figure 2·3. After sending the generated circuit to the receiver, the sender utilizes a method known as Oblivious Transfer to pass their input selections without revealing any of their private data to the receiver. The receiver requests labels from the sender for selection of their own inputs. During evaluation all labels from sender and receiver are utilized as input selections and used to decrypt the output of each gate in sequential order. A successful evaluation of each gate results in either inputs to the next successive gate, or a representation of the final output.

Even with optimizations (Beaver et al., 1990; Naor et al., 1999; Kolesnikov and Schneider, 2008; Zahur et al., 2015; Yakoubov, 2017), Garbled Circuits have a small

number of communication rounds but a large communication size (80-128× the size of the original data), rendering them beneficial in high-latency scenarios but detrimental when processing large datasets. However, GCs are costly in terms of bandwidth because the size of the encrypted representation is such that each gate is a multiple of the number of bits of the chosen security parameter. Using AES-128 where key length $\kappa = 128$, a single two-input gate would be $4 \times \kappa$ bits. There exist optimizations (Kolesnikov and Schneider, 2008; Beaver et al., 1990; Naor et al., 1999; Yakoubov, 2017) to reduce the gate size but the trend holds that each gate in a GC requires more bits to be communicated per gate required in the full function. This limits the maximum throughput of this scheme even when used in an ideal network with low latency.

Due to the size of each circuit and low number of communication rounds, Garbled Circuits is considered low in communication and high in computation. As the size of circuits increase, the amount of data required to send the circuit from one participant to the other increase. Thus, garbled circuits requires large bandwidth connections, but only required during initial transfer and communication is limited after evaluation begins. Method to improve upon the computation in garbled circuits is a heavily researched topic including methods of using co-processor accelerators in the form of GPUs and FPGAs for improved circuit evaluation (Hussain and Koushanfar, 2019; Fang et al., 2017; Fang et al., 2019; Huang et al., 2019; Songhori et al., 2015; Leeser et al., 2019).

Secret Sharing

Another choice for MPC is known as Secret Sharing. The general form of this protocol is usable for 'N' number of parties, this means more than two parties can perform joint calculations of secure data. Unlike garbled circuits, parties involved in secret sharing are an equal to each other and perform all calculations in lock step for a successful

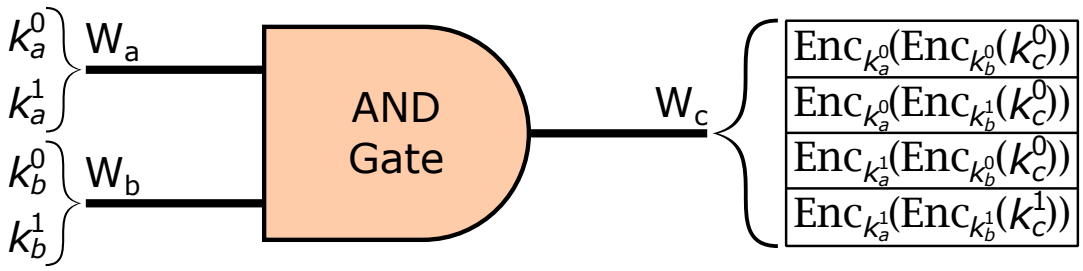


Figure 2·3: Each garbled circuit gate is composed of two input wires in the form of keys and an output wire. The inputs are selected through the process of oblivious transfer which hides the corresponding value from the evaluator. The output wire is a set of four values which are encrypted with the input keys. After input keys are selected, the output is decrypted and passed along to the next gate.

secure computation. Data is initial broken up into pieces known as 'shares' which are passed around to each party. Individual parties calculate the shared function requiring no additional knowledge during basic calculations until party members must communicate obfuscated pieces of their shares. Once final share outputs are generated and party members reveal the final value through one last round of communication between members.

Comparing against garbled circuits, secret sharing requires many more rounds of communication, however bandwidth per calculated gate is minimal. For example, many protocols performing a simple AND gate operation may only require one bit of communication per AND gate processed between two party members. Successive operations require each round of communication to complete before the next gate, thus latency per communication becomes the largest factor of performance when relying on secret sharing techniques.

We are are most interested in secret sharing protocols which require an honest majority. This means the calculation will complete successfully if all parties remain honest, with no data leakage if a single malicious party intervenes. Parties with more trust between members can choose to use a semi-honest protocol which allows for

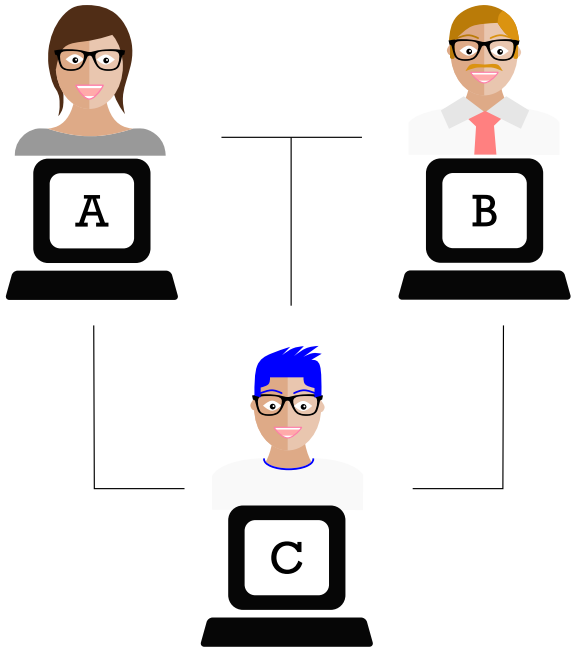


Figure 2·4: Three Party Multi-Party Computation

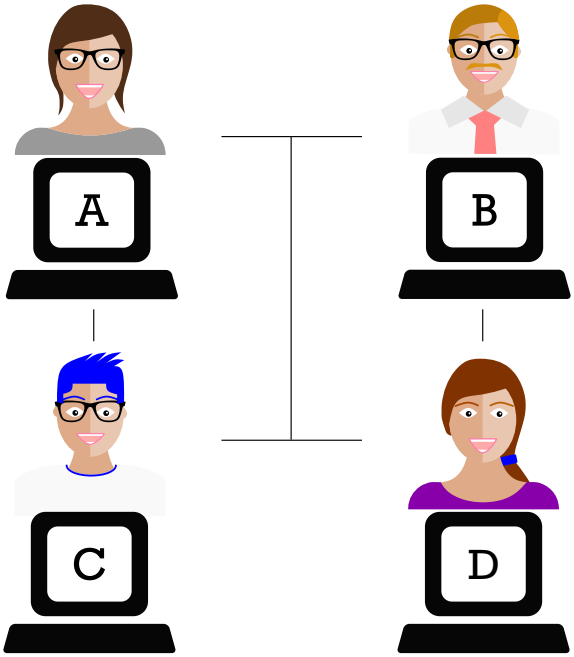


Figure 2·5: Four Party Multi-Party Computation

higher performance at the cost of security. In a semi-honest system, protections for malicious adversaries are limited or removed from computation process reducing the overall amount of work required for any application. Our original work focuses on a semi-honest 3PC using the Araki et al. (Araki et al., 2016b) method of secret sharing. This method creates and distributes 2 out of 3 shares of data between the 3 party members. This means only 2 cooperating party members are required to reconstruct the plain-text value of the final calculations. A more detailed description of this protocol can be found in section 6.2.

Our follow-up work reconstructs the algorithms for a honest majority 4PC using Fantastic Four (Dalskov et al., 2021). Shares are distributed in a 3 out of 4 member system, each member contains information for 3 shares and must communicate with one other participant to reconstruct each value. Malicious protection is done through a 'checks and balance' system during computation which requires additional communication rounds. Each party member first runs the protocol as expected and receives shares of data as intended. A second communication round receives hashed versions of the same data from a different party member. Next the receiving party performs the same hashing function on the original data and verifies that the calculated hash and the received hash value match. This step verifies that the data shared between members is accurate and each party member is lawfully running the protocol. We go into more detail for this protocol in section 6.2.

Chapter 3

The Data-Centric Evolution

The growth of the datacenter continues to achieve higher computation performance through an evolution into a data-centric datacenter. Advancements in intelligent network fabric enables this evolution by separating general purpose compute from acceleration and storage. A data-centric datacenter results in a multitude of new applications and resources for a more dynamic datacenter experience. Leading this revolution is the introduction of data processing units (DPUs) and FPGA SmartNICs as endpoint connectivity hardware for improved performance and communication between datacenter resources.

In this chapter, we first describe past methods of datacenter improvements through the addition of dedicated network hardware. Next, we walk through the recent evolution of intelligent network devices starting from commercial NICs and going to exotic FPGA SmartNICs in section 3.2. We discuss some key applications of intelligent network fabric to improve the overall performance and security of the current datacenter in section 3.3. Finally, in subsection 3.3.1 we show how FPGAs are an ideal target for many different applications in a new data-centric datacenter and how they are used in prior research for enhanced computing in subsection 3.3.2.

3.1 Performance Limitations

Focus on compute-based datacenters incur performance limitations from other resources. This includes data access times for large data sets found elsewhere in the

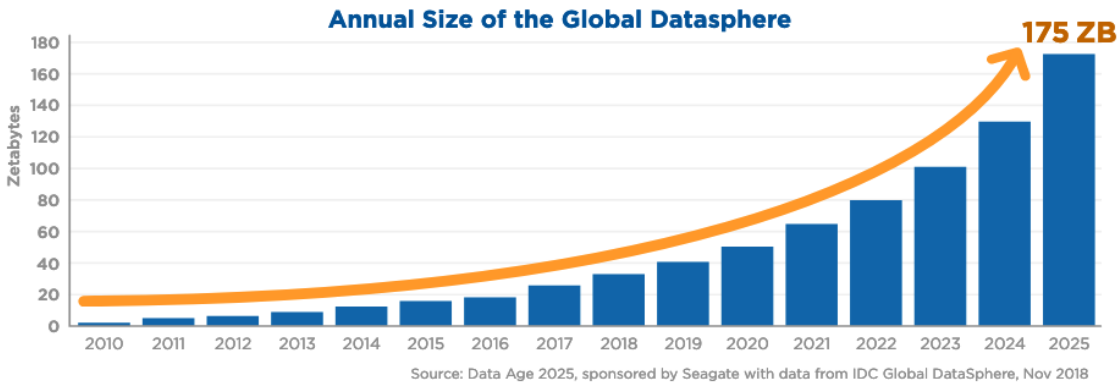


Figure 3·1: Approximation of global data collection (Reinsel et al., 2020).

datacenter. Big data applications range from multiple GBs, on average 14GB (Rowstron et al., 2012), to multiple Petabytes and growing. Datasets in the petabyte range cannot be loaded on a single bare metal instances and thus must be accessible through high performance distributed data systems.

As more applications rely on distributed compute and data access, the network begins to show the limitations of using traditional Ethernet network interface cards (NIC) with 1Gbps, 2.5Gbps, and 10Gbps. Low bandwidth networks are not sufficient enough for distributed big data applications. Advancements in both protocols and network communication technology lead to newer high bandwidth interface cards which can reach 100-200Gbps.

Higher network requirements in a distributed datacenter cause increased load on compute resources. Cloud providers must allocate more CPU cores to help with cloud infrastructure tasks and overall network management. Allocating cores per node for non-tenant applications reduce the number of available cores for virtual machine allocation and overall application performance. A breakthrough innovation led to the separation of the application space from cloud management. Separation of management operations such as hypervisor, security, and IO acceleration from node CPUs proved to be very beneficial to both tenants and cloud providers.

Amazon introduced Nitro in 2017 which replaced current AWS Hypervisors with a dedicated ASIC, Nitro card, controlling the hypervisor of each node (Amazon, 2022d). This removed the load on CPUs for tenants and helped to disaggregate the infrastructure layer from the application layer. Each tenant would only have access to application resources and additional resources would be available as part of a distributed network connected to the Nitro card system. After the initial release of Nitro, Amazon used the dedicated offload resources as a mechanism to provide a hard isolation between tenant applications and functions such as high-speed networking, security root of trust, distributed storage, and private enclave support.

Similar to Nitro, Microsoft showcased their use of FPGAs in a datacenter with Project Catapult (Caulfield et al., 2016). The initial Catapult design uses co-processor FPGAs connected on their own dedicated back end network to enable high bandwidth communication without interfering with datacenter network infrastructure. Catapult shows the capabilities of using co-processing FPGAs with high bandwidth and low latency dedicated communication interfaces to achieve distributed computation with Bing search improvements, specifically for the ranking service.

Microsoft then updated their initial design to Catapult v2 (Firestone et al., 2018; Li et al., 2016) which migrates the FPGAs from their own dedicated network to a Bump-in-the-Wire inline network connected accelerator. Each FPGA is between the NIC controller and top of rack switch (TOR). This provides complete hardware access to packets in process between NICs and enables a network of FPGAs which can communicate with each other without the need for host interaction, similar to Catapult v1. Utilizing this method of communication, they aimed to accelerate both applications and common software defined network functions which can improve the overall infrastructure of the datacenter. Microservices were designed as hardware only acceleration cores which can be utilized by any application to improve common

functions. Their showcase application Project Brainwave (Chung et al., 2018) helped to improve deep neural network (DNN) running on the Azure datacenter.

3.2 Classification of networking

Networking is becoming one of the most important factors in the datacenter as more functionality is being offloaded to discreet hardware. Not only do NICs provide communication between individual nodes, but they are being used to access all resources available in a datacenter. The increase in functionality of NICs has given rise to a new classification of different network hardware. Consumer grade traditional NICs typically run in the 1Gbps – 2.5Gbps range and require a host processor to perform most functions with packet processing. These have continued to be phased out for alternatives with higher performance as they provide ample communication bandwidth for most applications requiring large communication requirements.

We have seen the rise of 100Gbps to 200Gbps networks which are managed by dedicated hardware on the NIC and help to reduce the load of CPUs for network functions. The main difference between traditional NICs and high bandwidth variants are inclusion of dedicated offload hardware to eliviate the CPU requirements. Without this offload, CPUs must dedicate at minimum a single core for full-time network processing and may not handle the full throughput available, thus turning communication into a compute bound problem.

The introduction of SmartNICs utilize a combination of high bandwidth NICs and dedicated ASICs for hardware offload solutions. The dedicated ASIC hardware provide additional features such as hardened encryption cores, e.g. XTS-AES, remote direct memory access (RDMA), remote storage access, e.g. NVMe-oF, and virtualization networking like single root I/O virtualization (SR-IOV) and software defined networking (SDN) (NVIDIA, 2022a; Netronome, 2022).

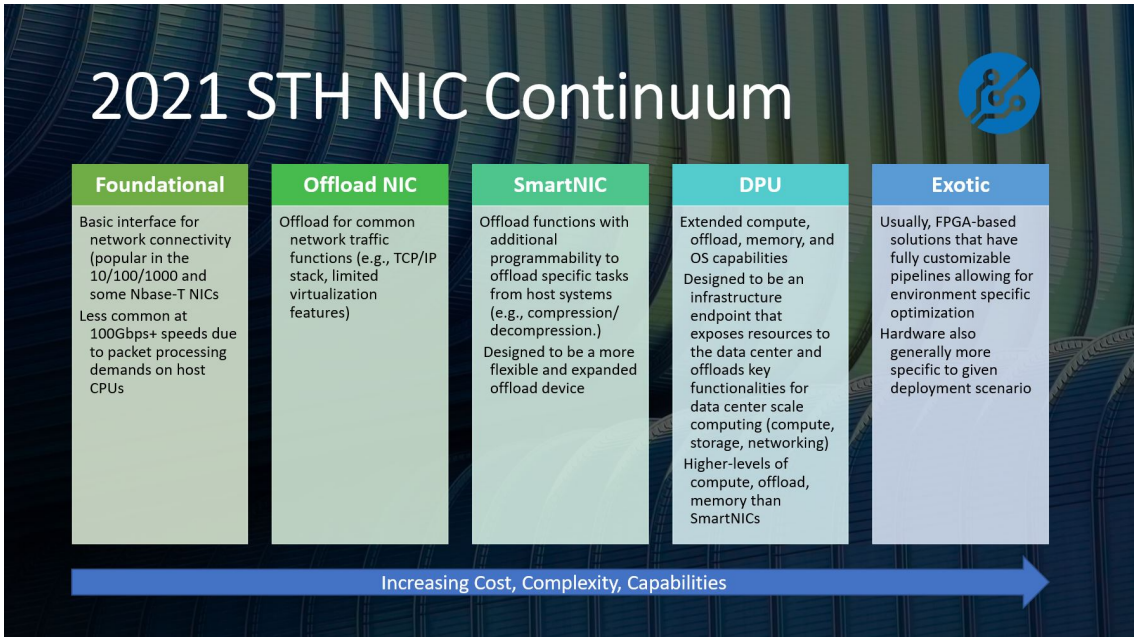


Figure 3·2: Classification of network interface cards in the datacenter environment (Kennedy, 2021).

Advancements in SmartNICs have led to Data Processing Units (DPU) which typically contain on board processors which can run full operating systems and enable a full software access to the flow of packets (NVIDIA, 2022b; Intel, 2022b). This enables full programmability of offload options to enable any additional functionality outside of the host system. We go into further detail in the following section.

Similar to DPUs, the FPGA SmartNIC enables reprogrammable hardware options to perform any packet processing function in-line with packets (Intel, 2022a; Cisco, 2022). As discussed earlier, the Microsoft Project Catapult utilized FPGAs with SmartNIC functions to program dedicated hardware microservices to accelerate network functions and large distributed computation.

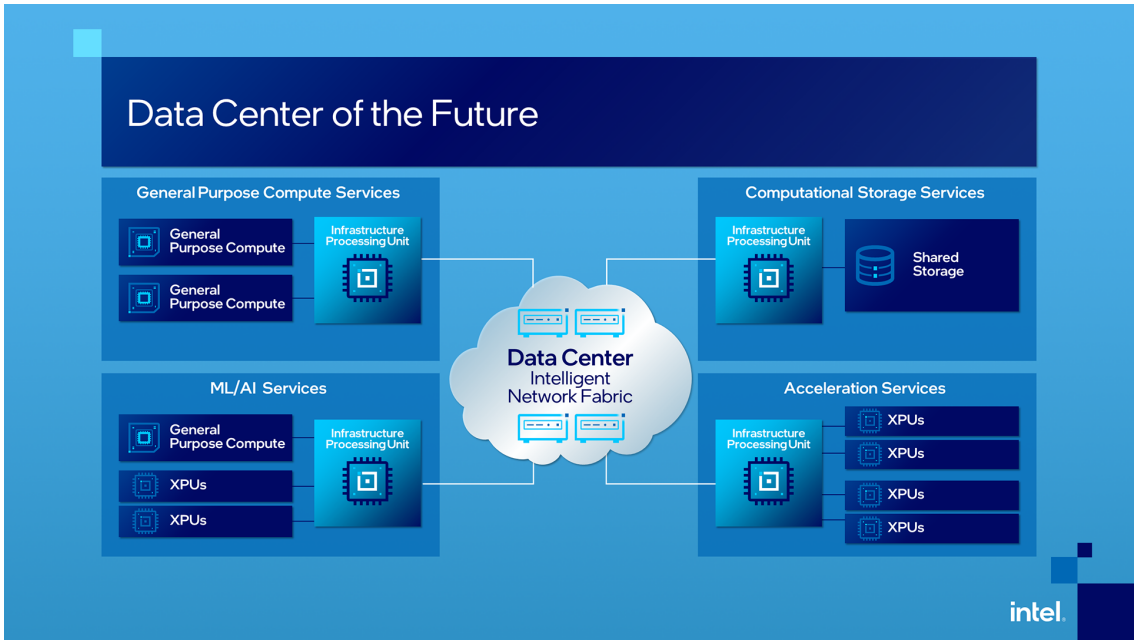


Figure 3·3: Intel's future of the datacenter fully connected through intelligent network fabric (Intel, 2021).

3.3 The Rise of the Data-Centric Datacenter

The use of DPUs provides many advantages as compared to previous public cloud infrastructures due to their headless operation and programmability. The future of the data-centric datacenter is that of a intelligent network fabric which connects all resources together and performs autonomous resource management enabling a global resource pool for any application. Some examples include storage pools, GPU and FPGA accelerator racks, on demand high memory systems, and a shared network of compute modules both accessible directly by tenants or remotely for distributed applications.

Creating a separation between tenant's application layer and the cloud infrastructure enable DPUs to utilize their resources independently from cloud tenants and help create and manage a more dynamic experience using virtual machines and cloud nodes. The Amazon Nitro system discussed earlier was the start of this method of

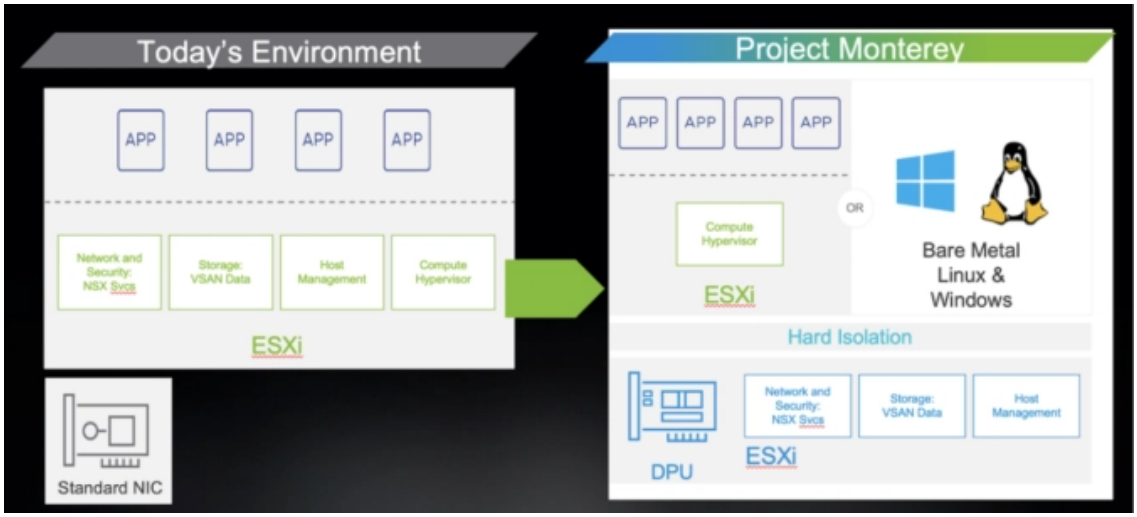


Figure 3·4: DPU integration into datacenter nodes provides both hardware isolation between application and management layers, and software offload for virtualization and network functions (Colbert, 2020).

separating the functions required by the cloud providers and applications that are utilized by tenants. With the initial Nitro card, nodes are managed by the cloud provider to help partition resources such as the network and accelerators as well as introduce cloud provider security features such as a root of trust and hardware attestation system. The network can be controlled through SDN options and even enable bandwidth limits and firewalls to prevent tenants from performing denial-of-service attacks internally in the datacenter.

In a data-centric environment individual resources are less often tied to individual specialized nodes as they were in the past, but now the resources are setup in a global model and access is given to any tenant. The usage of DPUs and FPGA SmartNICs enable a system of distributed resources which can be assigned and utilized at any given time through direct tenant requirements or distributed application use cases. Resources are tied to headless DPUs which control and manage the resource partitioning / time slicing, security, and additional data retrieval necessary to complete remote tasks. Utilizing protocols such as RDMA, NVMe-oF and SR-IOV, large pools

of resources such as storage are not required to be directly connected onboard a host CPU but are accessed over the networks and headless nodes of DPUs. Pools of accelerator resources such as GPUs and FPGAs are allocated through the connected DPUs and share each resource through time sliced or partitioning systems allowing for more accessibility options by all tenants of the cloud.

3.3.1 FPGAs in the datacenter

The biggest difference between DPUs and FPGA SmartNICs are the additional hardware options available using reconfigurable FPGA resources where DPUs with low power processing units are limited. FPGAs initially started as co-processor accelerators, but due to their high communication performance they have becoming a vital hardware resource in the growing data-centric datacenter. FPGAs directly connected through the network offer unique distributed accelerator opportunities where large applications can utilize the additional resources available with each FPGA. These accelerator networks are created with either dedicated FPGAs for application use, or rely on the additional resources available on FPGAs performing cloud infrastructure tasks. Figure 3·5 shows that network connected FPGAs fill the datacenter to perform many different tasks. FPGAs are utilized in many ways including:

- Co-processor - FPGAs used as a lookaside acceleration for specific application functions. This is similar to acceleration kernels in languages like OpenCL. Existing cloud vendors provide access to FPGA accelerators such as AWS F1 and Azure N-series.

- Bump-in-the-Wire (BitW) - FPGAs are placed inline with network controllers. Typically FPGA SmartNICs are the hardware of choice as they place FPGA compute resources between the NIC controller and onboard transceivers. This enables a large number of options for inline packet processing and application

acceleration without host interaction.

- Storage managers - Storage requirements in the datacenter continue to grow and devices such as PCIe based NVMe provide high performance storage options. Essential storage processing features such as virtualization, data encryption, and compression require compute resources and dedicated management controllers. FPGAs provide custom resources to offload storage functions and interpret interface protocols such as RDMA, RoCE, and NVMe-oF in a completely headless environment.

- In switch accelerators - Compute-in-the-network is a major field of study in distributed HPC applications as many communication bottlenecks are based on distributed network functions. Collectives in MPI are a large portion of performance overhead for many HPC applications. FPGAs appended to switches can help to minimize the congestion of running collective functions directly to host nodes. Advanced methods of MPI_Reduce require high software requirements to minimize network congestion during point-to-point communications. FPGAs attached to a switch can be used to intercept and redirect packets upon completing the collective operation to avoid any additional host software requirements.

- FPGA cluster - FPGA clusters enable distributed compute resources and remote accelerators with high bandwidth FPGA-to-FPGA communication networks. Each accelerator can be reconfigured with remote invocation of different functions, which enables a single host node to control any number of available FPGAs for large scale distributed applications.

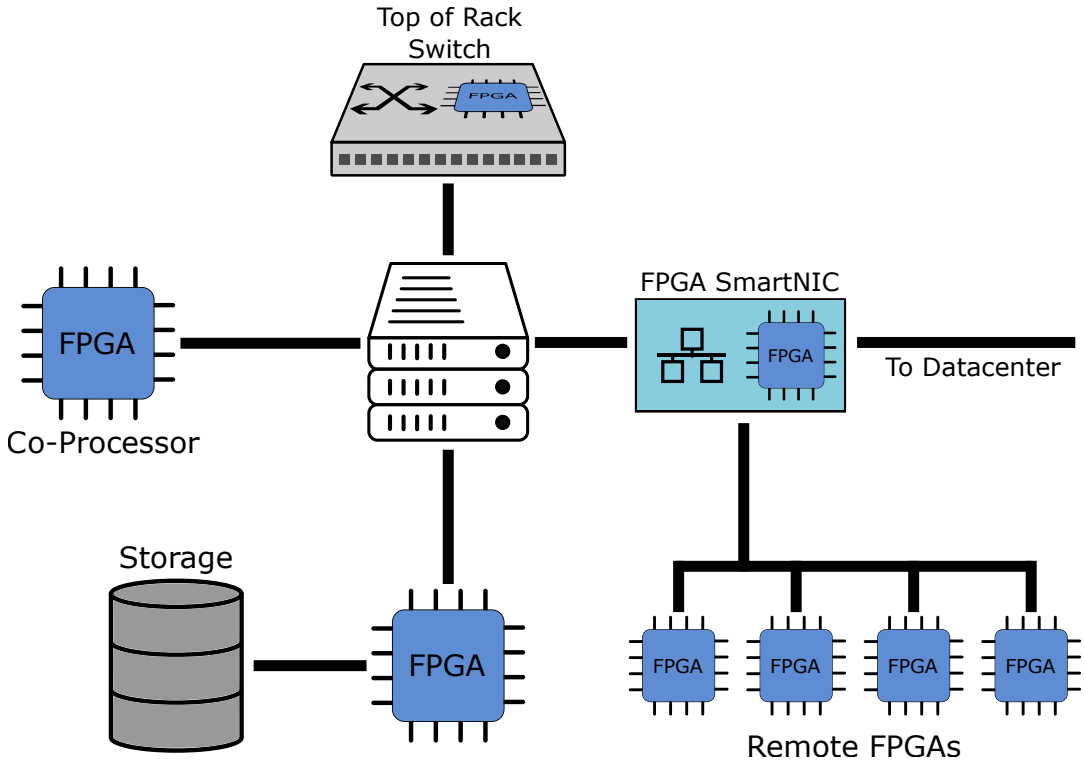


Figure 3·5: FPGAs are viable for many operations in the datacenter. Co-processor offload, remote headless accelerators, network attached storage controller, FPGA SmartNICs as Bump-in-the-Wire network accelerators, and top of rack switch collective acceleration are just a few.

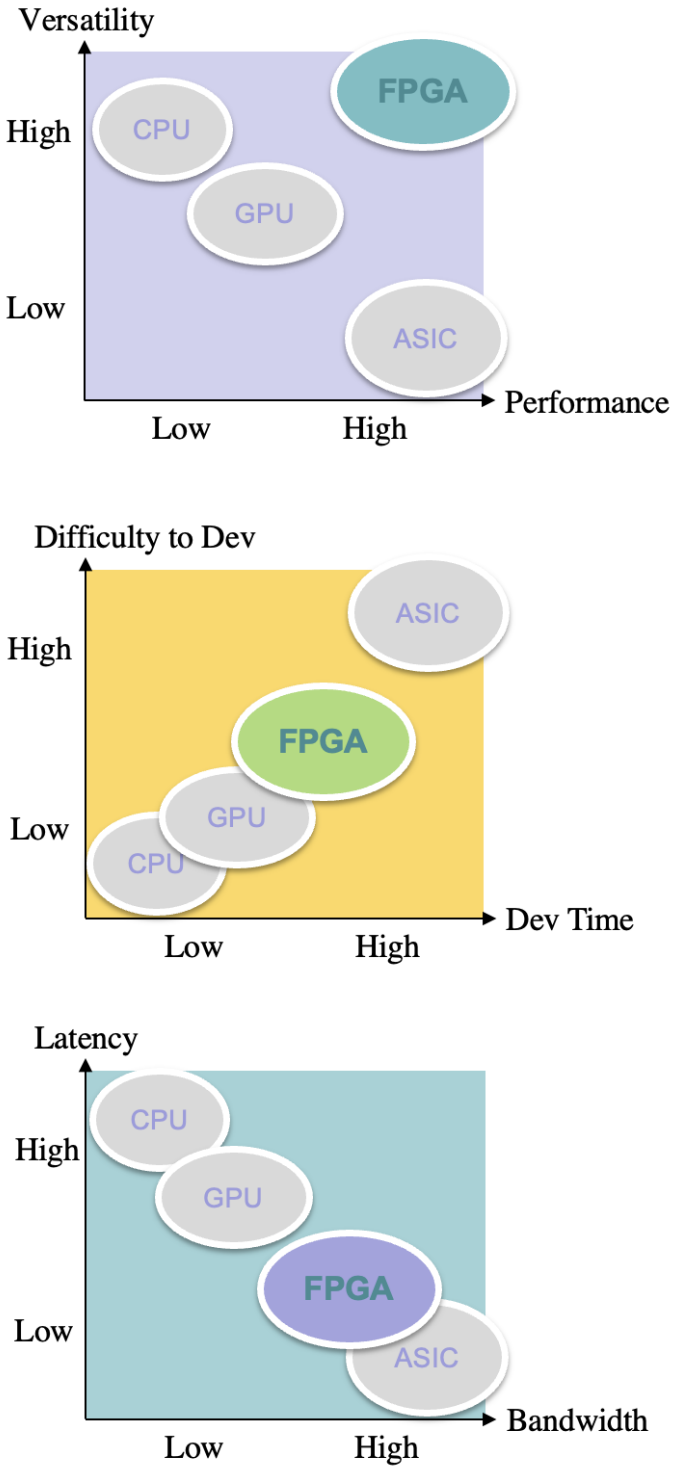


Figure 3·6: FPGA versatility and performance comparison between CPUs, GPUs, and ASICs.

3.3.2 FPGA Enhanced Computing

We leverage FPGA hardware as they provide ideal opportunities for application hardware acceleration shown in Figure 3·6. They offer great performance very close to dedicated hardware while still remaining versatile and reconfigurable. FPGA as communication processors support high bandwidth and low latency networks utilizing high performance transceivers. FPGAs offer a lower development time than their ASIC counterparts since they don't require the creation of custom silicon, but compared with CPU and GPU tool-flows, FPGAs are still more challenging to develop on. FPGAs are already available in the datacenter as acceleration offload engines (subsection 3.3.1).

Previous work utilizing FPGAs in HPC has demonstrated the effectiveness of reconfigurable hardware for many applications. This includes general overviews (Herbordt et al., 2007; Herbordt et al., 2008; VanCourt and Herbordt, 2009), basic work in programmability and performance (Herbordt and VanCourt, 2005; VanCourt and Herbordt, 2005; VanCourt and Herbordt, 2006a; VanCourt and Herbordt, 2006b), programmability and performance using a commercial tool chain (Yang et al., 2017; Sanaullah and Herbordt, 2017; Sanaullah and Herbordt, 2018b; Sanaullah and Herbordt, 2018c; Sanaullah and Herbordt, 2018a; Sanaullah et al., 2018a), FPGA system design and architecture (Khan and Herbordt, 2012; Sheng et al., 2015; Sheng et al., 2016; Sheng et al., 2017; Sheng et al., 2018b; Sheng et al., 2018a; Shahzad et al., 2021), FPGAs used with middleware such a MPI (Xiong et al., 2018b; Xiong et al., 2018a; Xiong et al., 2019; Xiong et al., 2020; Haghi et al., 2020b; Haghi et al., 2020a; Haghi et al., 2021; Haghi et al., 2022), and many case studies involving applications. In case studies, bioinformatics work includes studies of dynamic programming based algorithms (VanCourt and Herbordt, 2004; VanCourt and Herbordt, 2007), heuristic sequence alignment such as BLAST (Mahram and Herbordt, 2010; Mahram and Her-

bordt, 2015), multiple sequence alignment (Mahram and Herbordt, 2012), and other string matching applications (Conti et al., 2004). Molecular Dynamics studies include surveys (Herbordt, 2013; Khan et al., 2013), datapath optimization (Gu et al., 2008; Wu et al., 2021b), handling neighbor lists (Chiu and Herbordt, 2009; Chiu and Herbordt, 2010; Chiu et al., 2011), particle mapping (Sanaullah et al., 2016a; Sanaullah et al., 2016b), the long range force using multigrid (Gu and Herbordt, 2007), the 3D FFT (Humphries et al., 2014; Sheng et al., 2014), the bonded force (Xiong and Herbordt, 2017) and complete FPGA integration (Yang et al., 2019b; Yang et al., 2019a; Wu et al., 2021a). Other HPC applications include Discrete Molecular Dynamics (Model and Herbordt, 2007), Molecular Docking (Sukhwani and Herbordt, 2008; Sukhwani and Herbordt, 2010), Microarray Analysis (VanCourt et al., 2003), Adaptive Mesh Refinement (Wang et al., 2019b; Wang et al., 2019a), Multi-Party Computation (Wolfe et al., 2020; Patel et al., 2020; Patel et al., 2022b; Patel et al., 2022a), and Machine Learning (Sanaullah et al., 2018b; Geng et al., 2018b; Geng et al., 2018a; Geng et al., 2019b; Geng et al., 2019a; Geng et al., 2020a; Geng et al., 2020b; Geng et al., 2021a; Geng, 2021; Geng et al., 2021b; Geng et al., 2021c; Zhang et al., 2022).

3.4 Summary

As datacenters continue to evolve the use of DPUs, FPGAs, and FPGA SmartNICs have transformed the compute-centric datacenter into a data-centric environment leveraging high bandwidth and low latency networks. We show a solution of offloading networking functions with a full software-hardware toolkit and methods of using these FPGA SmartNICs for security-related applications, e.g., to help enable classified data processing without sacrificing confidentiality with the public cloud.

Chapter 4

Network Function Offload with FPGAs

In the last chapter, we described the evolution of a data-centric approach where computation is spread out across the datacenter. The deployment of smart networks makes this approach a reality through increasingly capable hardware such as FPGA SmartNICs. However, there are many fundamental problems with incorporating FPGAs as a replacement to traditional network solutions. Currently FPGAs are more complex to program and adopt compared to traditional network solutions. Unlike standard NICs, they are not easily portable between applications and require considerable insight into the hardware, and often vendor tools as well, for optimal performance.

In this chapter, we describe our work in addressing key issues to facilitate the integration and adoption of future FPGAs as network attached accelerators. We believe overcoming the programmability hurdle will enable future adoption of FPGAs in the datacenter. Creating an agnostic production environment between hardware and software enables applications to be optimized for FPGAs without the complexity of vendor specific features. In this chapter, we leverage Intel's Configurable Network Protocol Accelerator (COPA) as our agnostic production environment. COPA aims to utilize open-source software libraries and generic hardware development code to provide high performance communication with commercial FPGA solutions. We utilize this hardware-software framework to achieve our goal of improved security for private tenants with the help of FPGAs on a smart network.

We demonstrate the capabilities of the COPA framework through the implementation of hardware fragmentation of large packet payloads and the integration of the lookaside accelerator feature set. The hardware fragmentation application shows the advantages of network packet offload in reducing CPU resources and improving the congestion of network throughput with an unreliable connection. The inclusion of the lookaside accelerator enables a multitude of user programmable applications with direct access to the COPA network.

4.1 FPGAs and COPA

Adoption of SmartNICs continues to increase as a means to accelerate network functions and offload packet processing tasks away from CPU resources (Le et al., 2017; Schonbein et al., 2019; Eran et al., 2019; Tork et al., 2020; Grant et al., 2020; Miano et al., 2019). Past avenues of research focus on ASIC-based SmartNICs that utilize general purpose cores for packet processing, typically ARM cores, however currently face challenges in adoption as they are limited by both processing speed and specialized hardware functions. In contrast, SmartNICs with integrated FPGAs address these limitations by offering a reconfigurable hardware interface and enable inline acceleration of network functions. Utilizing FPGAs in conjunction with NIC offload processors these advanced SmartNICs improve upon the limitations through parallel processing and custom network functions.

Integrating FPGA SmartNICs into the datacenter should provide many advantages for both cloud providers and tenants, but similar to past FPGA adoption, the engineering effort required remains a large hurdle to overcome. Experienced hardware designers will utilize the FPGA as a blank slate and develop both the ‘shell’ required and application specifics to satisfy the application. Less experienced hardware designers will take advantage of the provided ‘shell’ as a starting point and

develop their applications directly utilizing the interfaces and communication channels through that system. This method is similar to deploying applications on the AWS F1 FPGA instances (Amazon, 2016). Amazon provides a general layout of all available communication functions to the host processor and allows for disconnecting of unused systems to save on fabric utilization to increase the available resources for larger applications. Unfortunatly, changes to the provided interfaces is completely dependent on Amazon and systems such as FPGA-to-FPGA communication still remains unavailable to this day. There is an additional hurdle for application designers as the implementation of the software layers which can natively communicate with these SmartNIC resources depends on the cloud providers defined interfaces.

The Intel Configurable Network Protocol Accelerator (COPA) (Krishnan et al., 2020; Krishnan et al., 2021) was developed to address these issues. COPA utilizes the open source software library, OpenFabric interface (OFI) libfabric (Grun et al., 2015), for platform-agnostic development and a standard for networking and acceleration invocation. In addition, the COPA hardware framework provides two options to reconfigurable accelerators, inline and lookaside, both of which are directly accessible from the libfabric API. COPA uses the on-board high speed transceivers, e.g., of the Intel Stratix 10 GX, and a uniquely designed architecture to enable high speed remote direct memory access (RDMA) between nodes at 100Gb/S line rate. Unique features include the ability for remote invocation of accelerators and headless operations for host free integration into a distributed datacenter environment.

COPA provides high throughput network connections running on existing Ethernet network technology thus making an ideal candidate to integrate as a direct replacement to existing NIC hardware. To achieve consistent 100Gb/S bandwidth, COPA utilizes techniques to offload software functionality directly into the COPA pipeline; examples include remote direct memory access (RDMA), inline packet pro-

cessing functions such as checksum calculation, and hardware level packet fragmentation. Each of the above examples help to reduce CPU requirements for network communication as the FPGA works as a dedicated communication processor and manages all packet traffic requirements. After deployment, as additional research and development is done the FPGAs can be reconfigured to include additional packet processing and software defined networking requirements to improve communication further without the need to invest both time and hardware into improved communication in the datacenter.

COPA also enables additional capability for tenants and cloud providers to utilize the remaining hardware fabric on the FPGA for application level hardware acceleration. This can be used similar to a dedicated accelerator co-processor or directly attached to the network as either an accelerator core or inline during packet flow. Accelerators communicating with the COPA network are useful for distributed computation applications such as those currently using MPI commands for data and control flow during processing. Any use case has the potential to boost the performance in both computation and communication through dedicated hardware accelerators.

We aim to show that FPGA SmartNICs are the most ideal hardware for the inevitable evolution of the data-centric datacenter; using the power of reconfigurable hardware to improve software defined network functions and application performance with high bandwidth and low latency networks.

4.2 Background

The COPA FPGA works by using the libfabric software API with included extensions to queue commands for processing by the hardware. Libfabric is part of the OpenFabrics Interfaces (OFI) (Grun et al., 2015) and is defined to enable a tight mapping between HPC applications and underlying networking device, known as fabric

hardware. Fabric hardware, using the OpenFabrics Interface standard, is co-designed with the alliance to provide an agnostic integration with software running the libfabric API. OpenFabric Interface is targeted at high-bandwidth, low-latency NICs, with a goal to scale to thousands of nodes. COPA's implementation of the libfabric interface creates a specialized command format to include both RDMA functionality and accelerator specific commands. Following the OFI provider API functionality, COPA utilizes dedicated message queues and event queues to provide control commands to the underlying hardware to perform PUT and GET functions with remote memory. These RDMA functions use the COPA network data paths to perform memory data transfer functions to host and onboard memory without CPU involvement. Previous work has shown the COPA network can achieve up to 100Gb/S bandwidth with zero-copy direct memory access (Krishnan et al., 2020; Krishnan et al., 2021). COPA aims to introduce network attached FPGAs into the datacenter with an open standard for programming and control of network hardware. Using an open standard would help to eliminate the engineering effort required for cloud providers when upgrading environments to new and improved fabric hardware. In addition, application developers can focus on the software layer abstraction for HPC applications and trust that the underlying hardware network will function without the need to understand the complete provider library interface.

Directly connected COPA device endpoints provide many usecases for distributed computation including database operations, machine learning at the edge, and sparse matrix operations. Applications with high data transfer requirements between endpoint locations map well to a distributed FPGA network and COPA provides an ideal interface to access and share memory point-to-point. Unlike DPUs with dedicated processing cores and full operating systems to provide additional headless control, COPA commands and dedicated hardware blocks create headless operations without

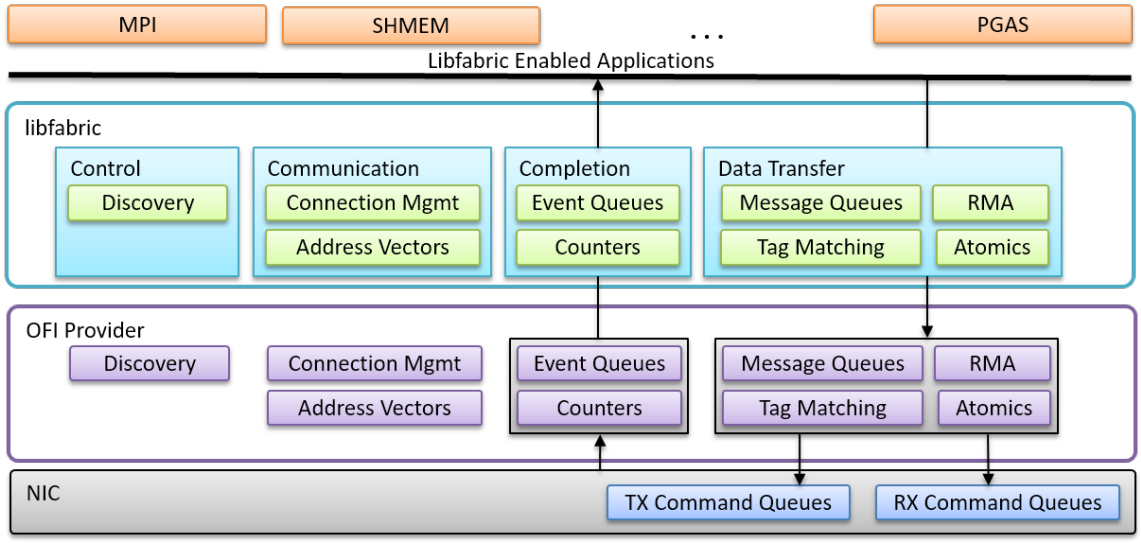


Figure 4·1: This diagram shows the overview of the openfabric interface or libfabric library. COPA is an OFI provider and has the necessary requirements to meet the software library API. Libfabric's purpose is the translation between high level HPC middlewares such as MPI and SHMEM into hardware networking devices in an agnostic manner. Hardware devices can utilize many forms of communication networks such as TCP, UDP, RDMA, and RoCE to name a few (Grun et al., 2015).

the need of a dedicated operating system. The combination of a FPGA COPA cluster and headless operation create an environment where a single host system can perform distributed computation and application across multiple FPGAs from a root location. Custom hardware acceleration and application integration open up opportunities for advanced distributed processing without the explicit need for host control over the cluster. We explore these future enhancements in section 4.4.

Hardware acceleration support comes in the form of inline and lookaside accelerators. Both accelerator options can be reconfigured by users for application specific functions.

- Inline - Inline functions operate on packets during transit in a bump-in-the-wire method, allowing for data manipulation of packets during egress or ingress of edge nodes. Two examples of inline acceleration are checksum calculation on data in transit or encode/decode functions with pre-shared key pairs. Multiple

inline accelerators can be toggled on and off and daisy chained on payload data.

- Lookaside - OFI commands can be constructed to trigger the lookaside accelerator functions both locally and on remote COPA nodes. Lookaside functions operate on data at rest, in host memory, and have direct connection with the COPA network to perform additional RDMA functions. Any number of RDMA COPA commands can be generated and placed into a dedicated hardware accelerator command queue for full COPA functionality. This enables the lookaside accelerator to perform data transfer, remote invocations, and memory alignment tasks without requiring the host to initiate network operations. Lookaside functionality is similar in concept to a dedicated processing core as found in DPUs.

We envision the usage of these accelerators in two methods, either developed in-house by application designers or available as an FPGA-as-a-Service system where accelerators are programmed before tenants request usage.

4.3 Architecture Implementation

4.3.1 RDMA functions

At the current time, COPA provides data transfer functions in the form of PUT (RDMA write), GET (RDMA read), and MESSAGE (send). Reliable and unreliable operations of each function is available with retries automatically generated at the hardware level of operation.

- PUT - Prior to data transfer, the remote destination address is advertised through libfabric support to be an approved location at the target location. The initiator creates a new command with the source address, advertised destination address, and length of payload data. Each command is placed into transmit command queue for completion. The COPA system takes each new

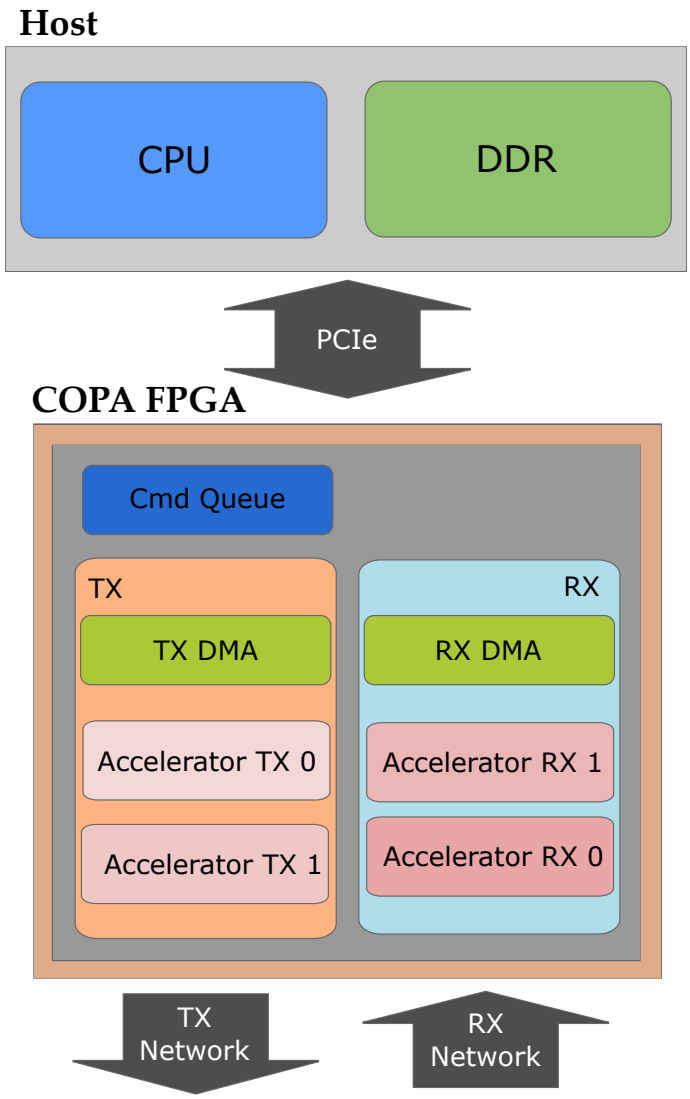


Figure 4·2: COPA Architecture with Inline Accelerator

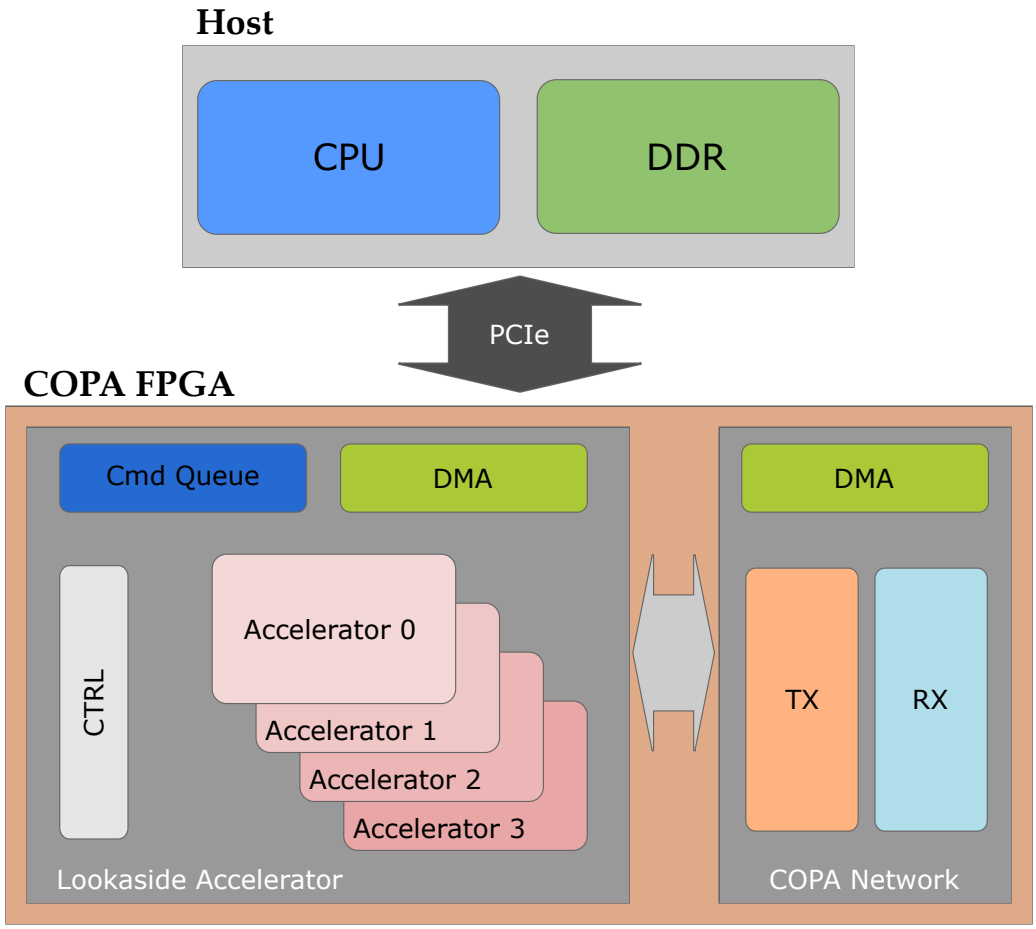


Figure 4·3: COPA architecture connected to host system through PCIe. The COPA FPGA contains a lookaside accelerator implementation directly connected to the COPA network allowing for network functions without host interaction.

incoming command and performs a DMA operation on the source address data from either the host memory or dedicated on-board memory and begins to construct a new packet with the destination the target memory address.

- GET - The initiator device requests data from a target location, which is then sent back to the initiator as a GET_RESPONSE packet. The source address is the remote data location in memory and the destination address is the initiator data location. Upon receiving the GET command packet on the target device, a GET_RESPONSE packet is created in a similar form to the PUT data packet with the device roles reversed.

- MESSAGE - Data being transferred does not require explicit buffer declaration, but instead data is transferred from initiator target buffer to the remote target buffer. Typically MESSAGE commands are used to send small payloads of data or configuration information.

- Unreliable - Upon completion of any command packet, unreliable operations continues to the next command in the transmit queue. This would be similar to UDP functionality in which dropped packets are understood by the application developer and should be considered plausible during normal operation.

- Reliable - Reliable operation however place each completed command into a pool to wait for expected responses from the target device. Response packets can be sent as standalone packets containing no additional data and only signaling acknowledgement of each successful packet id or piggyback with packets being sent from the original target to the original initiator. This piggyback method helps to eliminate additional packets on the network with a single acknowledgement bit of information.

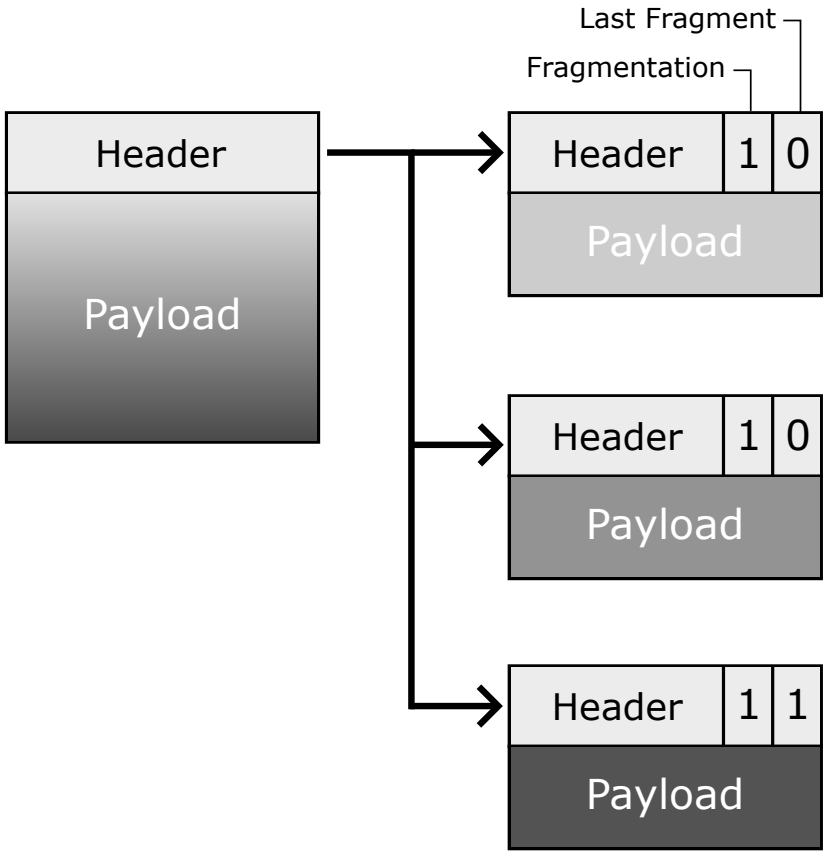


Figure 4·4: COPA Architecture Network Fragmentation Offload

Additional functions are being developed for more advanced functions in the form of remote invocation and bandwidth optimized functionality.

4.3.2 Network Offload - Fragmentation

Development of the COPA infrastructure continues to advance and provide more reliable high bandwidth operation of all network packets. Network function offload provides both less CPU utilization and enables saturation of available network bandwidth for all HPC communication heavy workloads. One such example is moving packet fragmentation away from the application and operating system layer into the NIC hardware. Some network resources may have a limit to packet payload size, also

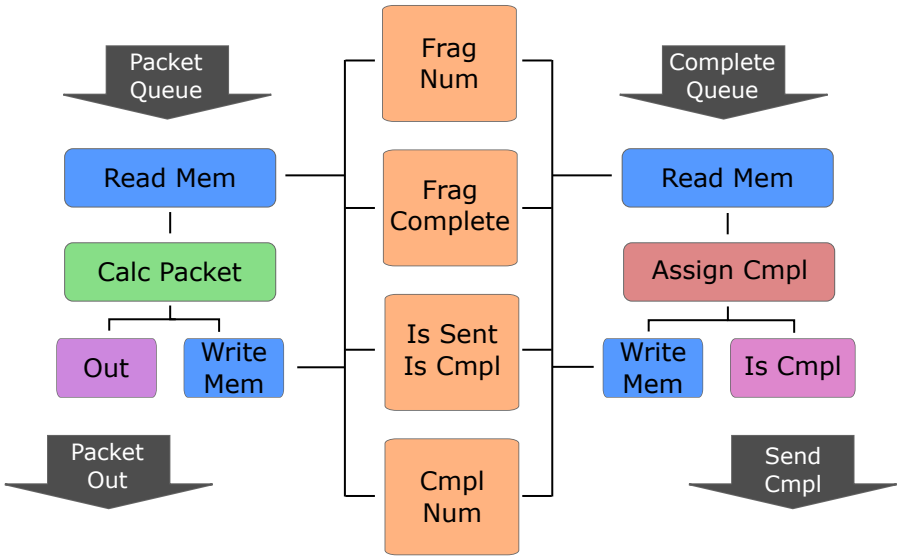


Figure 4·5: Hardware Fragmentation is setup into two flows with a set of shared on-chip memory for control and synchronization. The shared memory contains information for: Number of fragments required, Fragment currently being processed, Fragments sent and acknowledged, Total number of fragment completions.

known as maximum transmission unit (MTU), and typically perform packet fragmentation in software to indicate to the target node that the full data frame is spread across multiple packets. Reliable IP protocols such as TCP only send an acknowledgement after all fragmented packets are received for a single frame of data. If a single fragment is lost then all fragments for that particular frame must be transmitted again during retry phase. This can have very negative effects on bandwidth utilization and may cause network congestion if too often fragements are lost during transmission. Offloading fragmentation to the COPA hardware pipeline enables software to be agnostic to MTU requirements and packets can be added to the transfer queue with the largest MTU size available. COPA is then able to fill the transmit pipeline fully with fragmented packets, saturating the network and only performing retries for lost fragments without needing to resubmit entire packets again.

We demonstrate our implementation of hardware fragmentation in Figure 4·5.

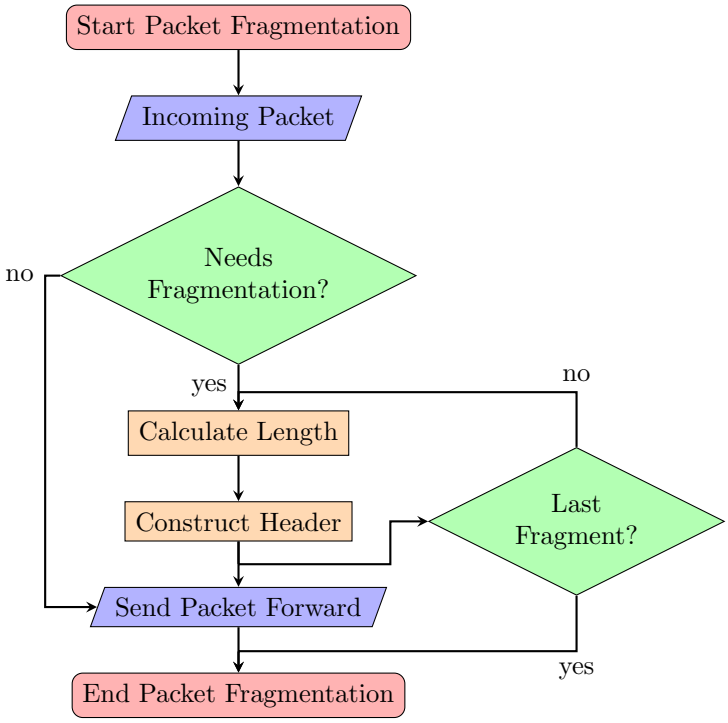


Figure 4·6: Flowchart of fragmentation process consists of determining if next packet requires fragmentation, calculation of new header and length information, and sending new fragment packets forward.

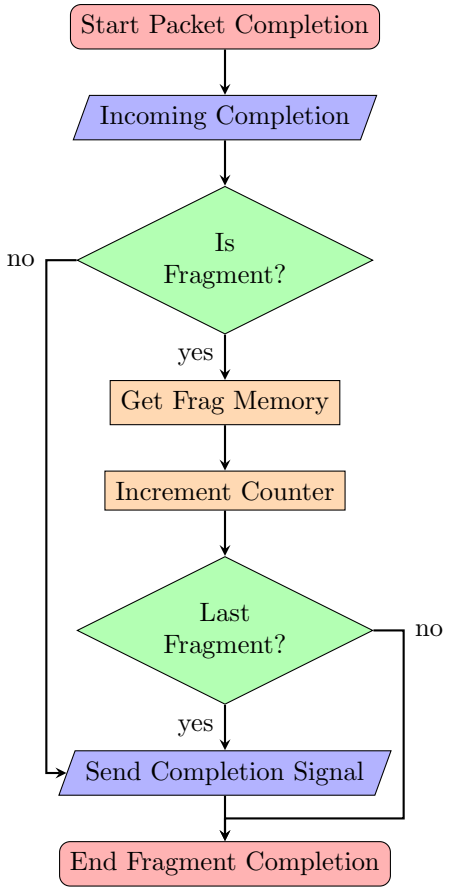


Figure 4·7: Flowchart of packet completion checks if packet was fragmented, updates the flag to indicate fragment completion, and sends completion signal on final fragment processed.

Our hardware fragmentation offload consists of two separate flows to control each packets current fragmentation status during operation in the COPA TX pipeline. To maintain concurrency, on-chip memory is used to store fragmentation information for both flows. The shared memory maintains information for each packet per processing queue and holds the required information for packet fragmentation. This information includes the number of fragments for each packet, the total fragments processed, the individual fragments completed successfully, and the total number of completion events for each packet. Using all this information synchronous behavior is maintained between the two flows.

The first flow determines the number of fragments required for the current packet. New header information is generated which calculates the new length, source, and destination address values. Each fragmentation packet behaves independently from this point forward in the COPA pipeline. The second flow is the completion flow which takes COPA acknowledgement signals and assigns them to the correct fragmentation portions for each individual packet. Each fragment must be completed successfully before the entire packet can be considered fully sent. Upon completion, a signal is sent up the pipeline to indicate the packet is fully processed.

Our results in Figure 4·8 demonstrate the throughput available with the included hardware fragmentation offload engine. These results are taken using unidirectional PUT commands with increasing sized packets from 32 bytes to 8 kilobytes. The fragmentation size is varied for each test. These results are similar to the original results (Krishnan et al., 2020), which indicates the inclusion of hardware fragmentation has negligible effect on the overall throughput of the system. In Figure 4·9 we present the improvement of the hardware fragmentation retry scenario vs traditional software fragmentation. As each fragment is considered individually, for the scenario presented, hardware fragmentation enables higher throughput for unstable

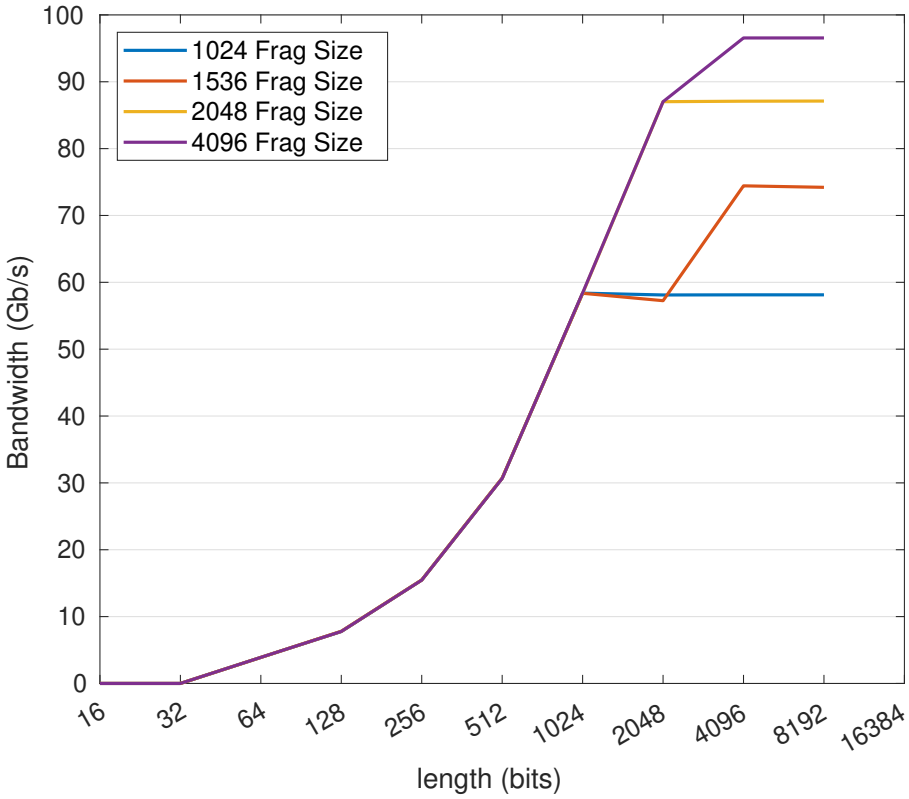


Figure 4·8: The inclusion of the hardware fragmentation accelerator show little change to the original COPA RDMA throughput.

connections. As presented, a drop rate of 5% for a system running on an 2kb MTU will only reduce the throughput by 4.35Gbps where software fragmentation of packets requiring at least 2 fragments reduces the throughput by at least 9.75Gbps.

4.3.3 Inline Acceleration

The inline accelerator provides methods of performing hardware operations on transmit or receive payload data. Function daisy chaining allows for data augmentation and additional operations on data stored in host memory. This is very useful for a number of applications which want to perform computation in the network. One example of daisy chaining inline acceleration is converting stored 32-bit integers into

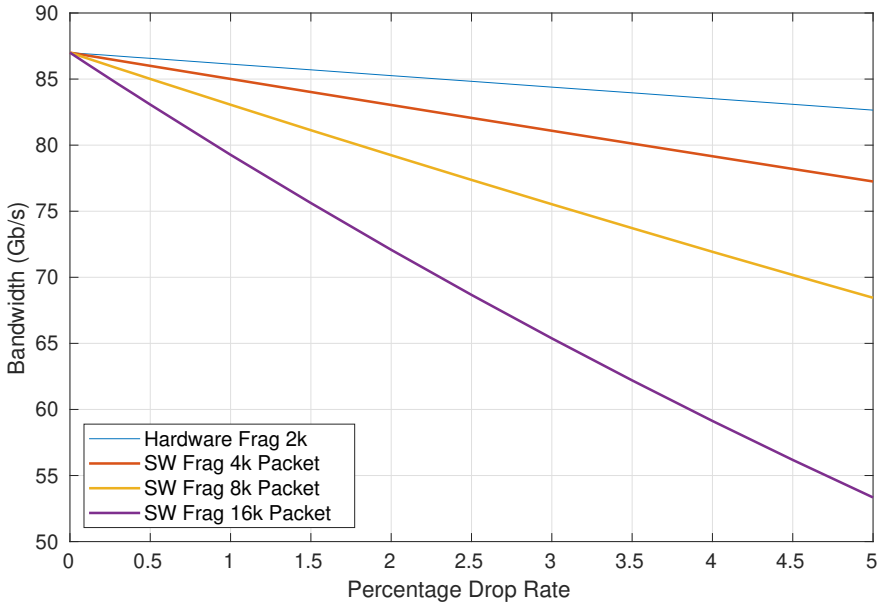


Figure 4·9: The graph shows a comparison between hardware fragmentation and software fragmentation with an MTU of 2kb. The test is done with increasing packet sizes and displays the drop in throughput for software fragmentation with an increasing drop rate. A saturated connection of 8kb with a drop rate of 4.5% will reduce software fragmentation throughput from 87Gbps down to 70Gbps, whereas with hardware fragmentation the total throughput is 83Gbps.

32-bit floating point and then performing checksum calculations on the payload data prior to sending forward to the target device. Checksum calculation is important for verifying correct data transmission, however is only valid on the correct data transferred. Thus performing data augmentation prior to transfer such as number conversion, the checksum should only be accurate utilizing the correctly converted values of the payload data.

Operations performed on payload data will introduce increased latency on the network, however due to the pipeline functionality the latency will remain consistent and minimal on successive packets such as for larger frames with fragmented data packets.

Additional examples of joint TX and RX inline operations are:

- encryption and decryption - Prior to network operations a private and public key or symmetric key can be shared between endpoints for packet security. All commands then after can indicate the usage of the encryption and decryption inline accelerator which would automatically perform encryption on transmission and decryption on packet reception utilizing the stored keys. Since this is done in the hardware pipeline, no data will be transmitted without going through the encryption process and successive retries would utilize the same encryption process but may contain uniquely different data in the payload depending on the encryption algorithm in use.

- compression and decompression - Utilizing state of the art compression techniques on streaming data, such as SZ (Xiong et al., 2019), payload size and overall network bandwidth can be reduced at the cost of compression and decompress latency. For large HPC applications such as timescale scientific data collection, real time data is less important as is collecting the most data possible and storing for later analysis. Inline compression enables data collection

initiators to send larger data counts with minimized payload sizes.

- checksum and verification - Similar to what was described above, checksum computation is important in maintain integrity of data transmission. Utilizing these function as inline accelerators allows for automatic data checking at the hardware network layer and can request for a retry in payload data if verification failed on the receiving end.

4.3.4 Lookaside Accelerator

The COPA lookaside architecture uses separate accelerator logic outside of the COPA network as seen in Figure 4·3. Acceleration is initially controlled by the host through a unique OFI command containing the source data location, destination location, length of data, and type of operation. A global control unit manages incoming commands in the queue and assigns them to appropriate accelerators. This feature enables a single host to issue commands to different accelerators for added parallelism or unique functions. The lookaside accelerator has full access to the COPA network through a dedicated hardware only command queue. Each accelerator initially collects the source data through a DMA operation from the host memory. If source data is unavailable locally, i.e., found on a remote COPA node, then a COPA network command is generated and used to obtain data from the remote node prior to DMA operation. Following DMA completion, acceleration is performed and final calculations are sent back to memory through a second DMA operation. If the destination memory location is for a remote node, then the COPA network is again used to send the final completed values to another COPA node on the network. To demonstrate this functionality we utilize our MPC application as a perfect candidate to gain both computation and communication improvements with COPA FPGAs, details about our implementation can be found in section 6.1.

Functionality of the lookaside accelerator have many different use cases for advanced application requirements. An FPGA network running COPA can take advantage of the large amount of compute nodes available and perform remote acceleration near data storage. An ideal scenario would be for database key-value store operations where data is located on target NVMe devices directly connected to PCIe channels to the FPGA. This is similar to NVME-oF but instead of transferring all data back to the host processor for computation, remote accelerators can perform data operations at the target storage location and only send response data based on application requirements. This helps to reduce congestion on the host network and enables a simple distributed application network configuration.

If processing and data storage are decoupled from one another, distributed applications can gain advantages of the network resources by performing distributed computation on a remote cluster of COPA nodes with each node requesting and submitting data to off-device storage without host knowledge. An ideal usecase would be for applications such as molecular dynamics (Stewart et al., 2021) where computation can be performed in parallel utilizing many accelerator nodes, but do not necessarily contain all the information required on local memory. Requests for remote memory can be performed in unison only when required without host intervention enabling high performance parallel operation for each iteration of the simulation space.

4.4 Future Enhancements

Even after adoption of FPGA SmartNICs in the datacenter, future enhancements can provide additional benefits for all types of applications and only require a simple reprogramming of the FPGA. A few examples of enhancements include:

- Scatter Gather Memory Access - Hardware optimized memory access patterns replace CPU buffer pack and unpack operations, reduce the total number of

packets on the network, and improving overall payload size (Gainaru et al., 2016). The usage of larger payloads improve the bandwidth utilization of the network, but can cause increased congestion with errors and dropped packets. Finding a balance between scatter-gather memory access is important to improving the performance of RDMA memory applications and overall communication requirements.

- Small Message Coalescing - Currently utilizing the full 100Gb/s bandwidth requires large payload sizes per frame of data. This means less bandwidth is used with packet headers allowing for more payload data at any given time. Applications with smaller more frequent parallel payloads can benefit from higher bandwidth networks through coalescing of small packets into a single larger packet with the same destination. The scatter operation can help directly allow for none contiguous memory patterns to be combined into a single packet. An advanced mechanism would allow for data to piggyback on packets with intermediate destination hops but would require a much better understanding of the network topology in the system.

- Remote Atomics - Distributed applications suffer from synchronous operation hurdles where large distributed applications must be performed on a number of servers with knowledge that each server is in lockstep. Enabling a feature to maintain synchronicity between nodes can provide useful to allow the host controller to assume all clients are operating together and do not require additional programming effort and bottlenecks with barriers and other synchronous checks.

4.5 Summary

In this chapter, we demonstrated that an agnostic production environment such as COPA provides a myriad of network acceleration opportunities with an open-source software interface and a generic hardware development framework. In the following chapters, we utilize the custom features available with COPA as a platform to enable unique security opportunities through private cloud enclaves and high performance MPC. The inclusion of the hardware fragmentation offload engine demonstrates a completely host-independent functionality of packet processing. In the upcoming chapter we utilize this independent behavior to introduce a two layer network isolation approach. The inclusion of the lookaside accelerator enables custom applications to gain performance through both improved computation and communication with network connected FPGAs. The lookaside accelerator features are utilized in chapter 6 to enhance Multi-Party Computation.

Chapter 5

Secure Public Cloud Networks for Classified Environments

Diverse institutions such as government, healthcare, and financial organizations all have very specific classified data requirements, but will continue to face challenges with growth of hardware resources as they depend on a high level of security guarantees. To maintain security and privacy of their confidential data, these organizations commonly utilize private on-site datacenters or trust private cloud vendors with maintaining a small physically isolated resource pool as a short-term solution. Even with this temporary solution one problem exists: maintaining secure and reliable communication with organizations and individuals outside of the physically isolated environment. We demonstrate a method of using a network attached FPGA to reduce the potential attack surface of man-in-the-middle and network snooping attacks for current and prospective vulnerabilities.

5.1 Threat Model

Organizations rely on public resources as a solution to provide agile and dynamic resource updates for ongoing improvements to current performance bound applications. Cost and maintenance of private on-site datacenters grow at alarming rates as organizations attempt to stay competitive while maintaining high levels of privacy over their confidential data. This is why public clouds are considered an opportune next step to enable a relatively low cost of entry and provide a reliable environment

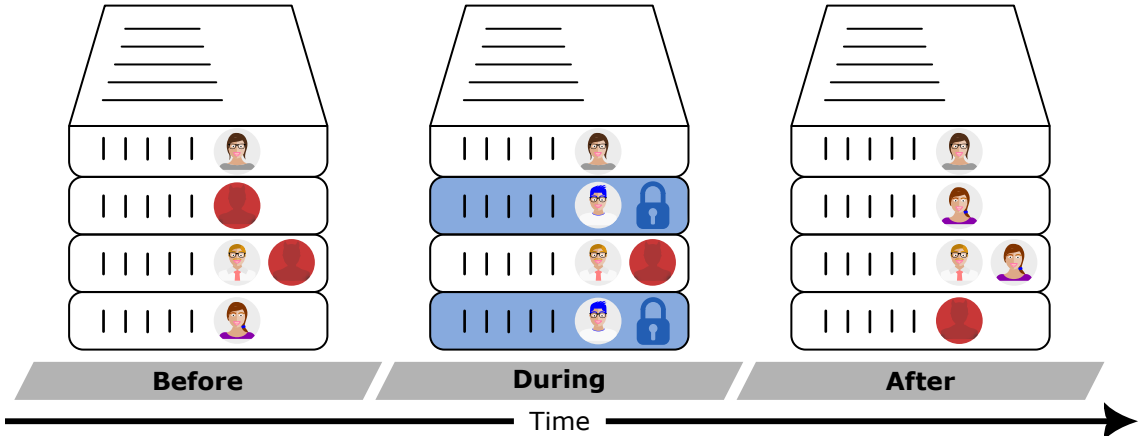


Figure 5·1: Malicious actors have many opportunities to steal classified information through both spatial and temporal attack vectors. Spatial relies on having access to the nodes adjacent to the classified provisioned hardware during use. Temporal is the access of classified nodes before or after use.

for any organization looking to utilize the newest hardware advancements and large resource pools.

The possibility for dynamic private cloud enclaves in a public cloud setting would reduce the cost of deployment and maintenance of in-house private datacenters. Borrowing public cloud resources for private enclaves enable dynamic addition of on-demand cloud nodes when requested. However, the attack surface of shared public cloud nodes would increase as security in the datacenter would require both a spatial and temporal understanding of malicious adversaries with shared hardware resources. Figure 5·1 demonstrates an example scenario of potential attack vectors with shared resources as both spatial and temporal attacks can result in a loss of privacy to private cloud nodes.

A spatial attack vector is shown through the sharing of nodes provisioned for classified organizations. Research has shown that tenants who obtain access to the same physical node while utilizing a separate virtual machine (VM) can obtain classified

data through side channels such as cache, network, and memory (Yarom and Falkner, 2014; Liu et al., 2015; Gu et al., 2018; Ling et al., 2012; Lindemann and Fischer, 2018; Hund et al., 2013). It is important for nodes which are allocated to classified processing be separated from all other tenants and only accessible by the authorized tenant. This is similar to services offered such as bare metal instances (Barr, 2017) where the whole node is provisioned removing any additional access through virtual machines.

A temporal attack vector is shown through the reuse of nodes with other tenants in the cloud, both before and after classified organization provisioning. As this is a public cloud environment, any node which is requested by classified organizations could be available for public cloud tenants as a usable resource. A requirement for classified nodes is to enable an attestation mechanism or secure provisioning and decommissioning of each node which guards against attackers. Malicious individuals may leave harmful code running on a node prior to classified usage to obtain information as another tenant utilizes the same node. Communication with harmful code can a form of covert channel, leaking data while the node is in use to an outside entity, either a different node in the same rack or outside the cloud entirely. Additionally, after a classified organization returns the node to the public pool of available machines, a malicious tenant may attempt to gain access to the same node and reverse engineer data from latent information remaining on the node.

5.2 Network Threats

So far, we have seen attacks to the shared nodes in both a spatial and temporal manner. Another shared resource that must be considered in our threat model is the network architecture for each private enclave. This is important to consider as remote attackers can either passively or actively reduce the confidentiality, isolation,

and assurance of the system.

Active attacks are those which aim to prevent the access of the node, compromise the integrity of the private cloud enclave, obtain full control of a private node, or any combination. Some examples include, man-in-the-middle attacks, Denial of Service (DoS) attacks, MAC and IP address spoofing, VLAN hopping, and many more. The goal of the malicious party can be to limit the confidentiality of the system and obtain classified information, limit or remove the isolation through gaining access to nodes or redirecting data outside of the private enclave, and / or remove the assurance the system is available at any time especially during time critical events. Once an active attack is performed the victim node may be completely inaccessible and all data lost to the hands of the malicious party.

Passive attacks are those which try to obtain confidential data through monitoring or performing side channel or covert channel attacks without disrupting the functionality of the private enclave node. Theoretical attacks can range from monitoring network packets through a shared top-of-rack switch, or obtaining access to network configuration features which can route packets from a private enclave node through a man-in-the-middle device before reaching the intended target node. These examples do not give the victim any indication that the node is compromised and reduce both the confidentiality and isolation of the whole system.

One method to combat data leakage through network attacks such as snooping and man-in-the-middle attacks is to always use a form of encryption when accessing the public network. This is a very common approach to mitigate attacks trying to learn confidential information from plain-text network packets. RSA encryption is still the most widely used public key encryption method but has seen some backlash as private keys generated with poor prime number selection can reduce the security of the algorithm (Lenstra et al., 2012; Shumow, 2020). This gives malicious individuals a

far easier approach to retrieve the private key information. The ability to dynamically update and change the encryption methods help to combat future attacks against ageing encryption engines.

Government restricted classified data rely on special protection systems which are certified by the national security agency (NSA). One example of specialized hardware encryption components are known as Type 1 encryptors (Smid and Branstad, 1988), e.g. High Assurance Internet Protocol Encryptor (HAIPE) (NIST, 2022). Type 1 encryptors are certified products by the NSA to be used for securing classified data such as U.S. government information. The internal encryption algorithms can be based on well-known protocols, internet protocol security (IPSec), but contain additional information and safeguards to ensure the device is not tampered with or compromised. Protections include both physical and remote attack vectors after deployment or during operation. Due to the rigorous certification process, Type 1 encryption devices continue to be costly for individual deployment and are rarely updated. This may result in future threat vectors as compromised encryption processes or newly developed attacks can force agencies to offline their current network as they wait for updated security devices to be available.

5.3 Commercial Solutions for Classified

Designing an approach to enable protection against all described threats is a monumental task. However, there are guidelines created for government entities which enable usage of commercial hardware and resources known as Commercial Solutions for Classified (CSfC). The current methods of protection for classified use cases are to increase the number of security layers in a system to combat a single compromised layer. We will focus on those guidelines which are most relevant to the private cloud enclave use case.

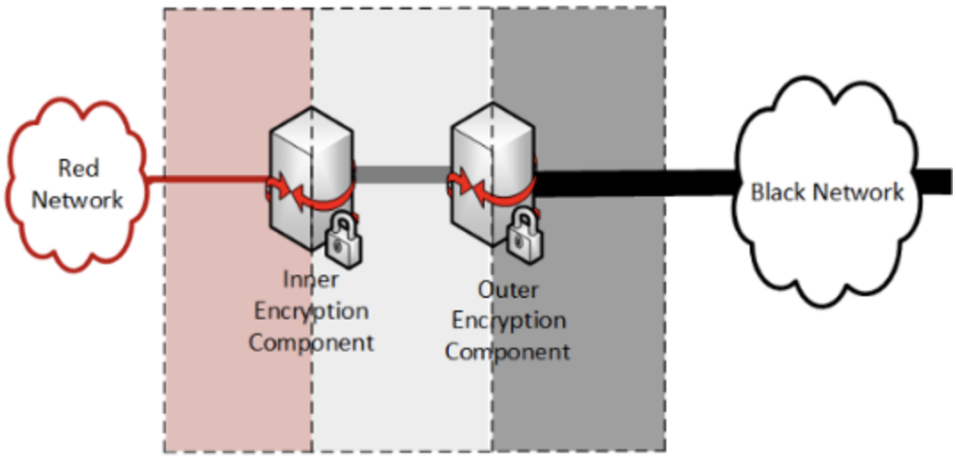


Figure 5·2: CSfC Network Requirements

The goal for the CSfC guidelines is to enable a double protection layer for communication over a public untrusted network. In Figure 5·2 we demonstrate this through the use of two unique solutions for encryption of data when crossing network boundaries. Commercial Solutions for Classified incorporates multiple unique layers of protection to safeguard against malicious parties from compromising the system with only a single attack vector. This is useful against many different scenarios including manufacturing attacks before before deployment and remote attacks after deployment.

A large pool of available devices give opportunity for multiple hardware manufactures to create and deploy systems which are unique to each other. This is important as the manufacturing process of each encryptor is independent of each other and spreads out the chance for compromised processes entering into both layers of encryption. For example, if manufacture A creates the first layer Type 1 encryptor and manufacture B creates the second layer. A compromised device such as a hardware trojan during hardware fabrication of manufacture A would not have any impact on the manufacture B device, thus maintaining security of the system even after deployment.

Utilizing unique algorithmic deployment approaches of encryption protocols can also help mitigate compromise of a network environment. While two devices may

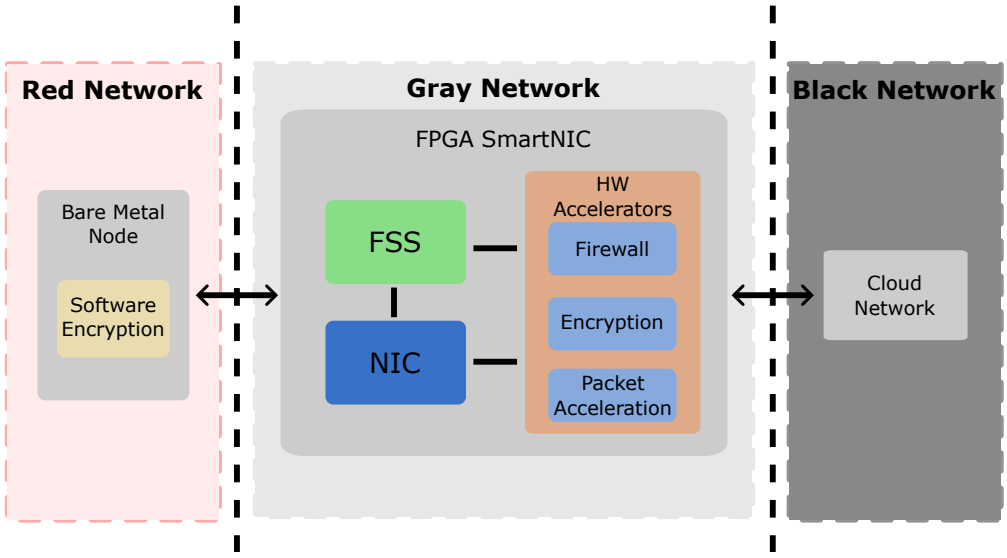


Figure 5·3: Classified network microservices on FPGA SmartNIC

both utilize IPSec for their security protocol of choice, each device may implement the algorithms differently, e.g. hardware ASIC, software on CPU, CPU co-processor accelerator. A single attack vector targeting a specific deployment approach may compromise a single device, but is ineffective against a second device using a different approach. An example of a compromised approach can be seen in the attacks against Intel SGX and the ability to retrieve private keys from a compromised system. This attack cannot be used similarly when attempting to obtain keys with a system using a unique hardware ASIC approach.

5.4 Cloud Network Attached FPGA Design

Incorporating Type 1 encryptors into a public cloud datacenter adds cost to each node and would require a large change to the current public cloud providers infrastructure. Network attached FPGAs can provide a unique solution to mimic the functionality of custom Type 1 encryptors and reduce the monetary and engineering cost for the cloud

vendor. Positioning FPGAs in a Bump-in-the-Wire manner, connected between the node and the outgoing network, no additional wiring would be required and default functionality would pass all packets through the FPGA system with no alteration. This would enable uninterruptible addition of network attached FPGAs without a major rework of the cloud infrastructure. Once incorporated, these FPGAs can provide many features for security, application specific acceleration, and cloud native functions.

Due to the dynamic hardware nature of FPGAs, all functions can be selectively utilized based on cloud vendor or tenant usage. Common network related functions including firewalls, packet monitoring and analysis, or even encryption and decryption are easily incorporated as the FPGA would function as a single port switch and route all packets from ingress to egress. The setup of each network function can be done dynamically by the tenant or statically through the cloud vendor interface.

We imagine the inclusion of specialized security kernels for tenant customization called Firmware Secure System (FSS). The goal of the FSS is to enable a secure trusted method of updating individual functions on the FPGA. Specialized encryption algorithms can be used in place of common methods with updated designs to combat current and future attacks. Bootstrapping keys at a unique update interval would help to limit the usability of stolen keys after a potentially successful attack by a malicious party.

5.5 Prototype Scenario

In our initial prototype design, we utilize a Xilinx Kintex KCU105 setup to utilize a MicroBlaze softcore embedded processor running Linux connected to the QSFP transceivers as an ethernet device driver. The FPGA device is placed between two devices simulating two cloud nodes on a public network. Each edge node has IPSec

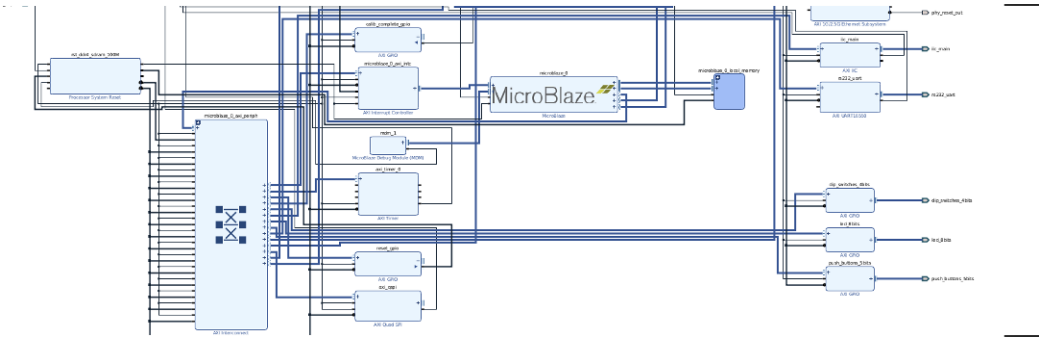


Figure 5·4: Initial Security BitW FPGA encryption design.

functionality available and encrypts all traffic between both devices. The FPGA contains a custom firewall and packet forwarding mechanism which is performed using IPTables in Linux. We envision this functionality in the future can be done using FPGA hardware resources without the need for a dedicated processing system.

Each host node generates packets as if performing normal communication tasks between two systems. They are then run through a software IPSec protocol to perform initial encryption on each packet before passing to the FPGA subsystem. The general process of the FPGA is to pass through all packets to their destination. Utilizing a Bump-in-the-Wire configuration packets can be blocked if necessary using a configurable firewall, and packet counts and statistics can be captured using a simple packet monitoring mechanic. Finally, accepted packets out of the FPGA are transferred onto the public network and wait processing by the corresponding FPGA on the destination interface.

Similar to Type 1 encryptors, each FPGA can be configured with specialized firewall parameters and unique private key generators. Each generator can cycle through private keys based on a set lookup table of approved destinations. This mechanism helps to limit attacks from gaining access to a private node as any successful attempt at stealing private keys would only be useful for a limited amount of time.

Network attached FPGAs provide unique functionality while attempting to improve the security of the network from side and covert channels attacks. With the

usage of the Microblaze software processing in our prototype, we can introduce mechanisms such as dummy packets or fixed size payloads. These examples are just a small addition to prevent malicious parties from gaining additional information through the means of a side channel attack.

5.6 Analysis

As we continue to see FPGA adoption for many different application scenarios, our prototype system shows a unique approach to providing additional network isolation techniques to privacy focused tenants. The main goal of the FPGA network security features is to provide tenants with a custom security approach while trusting cloud provider networks during secure enclave operations. Unlike traditional network security through Type 1 encryption devices and commercial hardware-based network solutions, the FPGA Bump-in-the-Wire method allows tenants to control the amount of security required through a tailored experience. The addition of custom encryption engines, firewalls, filters, and network monitors are just a few examples of accelerators to enable tenants full control over their network security while complying with standards similar to CSfC.

One example of customization is each tenant's ability to change the implementation for encryption engines. In theory, any algorithm can be adopted to the FPGA allowing for very unique encryption protocols for tenants such as the government. This enables government organizations to implement highly classified encryption schemes unknown to the public directly onto private enclave networks. This can remove the potential for adversaries to use future network encryption attacks on common protocols such as AES to decode messages during operation.

The use of public resources always poses a high risk situation where control over shared hardware can be compromised before, during, or after tenant use. Systems

Table 5.1: Network Isolation Comparison

	Cost	Physical Protection	Encryption	User Control
NSA Type 1	High	Yes	NSA Cert	Low
CSfC	Low	No	IPSec	Med
Private Cloud	Med	Yes	Multiple	Low
Private Enclave	Med	No	Software	Med
FPGA	Low	No	Custom	High

like the FSS can provide additional features to maintain security over the FPGA programming interface, or verify the functionality of the accelerators through attestation methods (Vliegen et al., 2019). It is important to verify that each FPGA is only using the tenant supplied functions and does not leave any confidential information on the FPGA once returned back to the public cloud. Attestation with the Firmware Secure System opens additional features for many important security requirements such as a hardware root of trust (Schear et al., 2016), private key management systems (Fornero et al., 2020), packet monitoring and control (Ghafir et al., 2016), and network concealment (Fraunholz et al., 2018).

There are trade-offs when using FPGAs for network isolation compared to existing products. Table 5.1 shows how the technology we previous discussed compares to our FPGA prototype. Based on customer security requirements, cost, and customization features, the FPGA isolation techniques provide an ideal system for communication between trusted nodes over untrusted networks. Customers which require the highest level of security over their private data will continue to utilize Type 1 encryption devices as they provide the highest network isolation guarantee while sacrificing cost and user control. However, if customers require elastic hardware environments, private clouds provide isolated pools of resources physically secluded from the public. We envision a combination of FPGA network isolation techniques with private enclaves would enable a method of trusted communication between private nodes and external

systems over the public cloud network. Using the numerous customization features, FPGAs would allow tenants to implement any number of isolation requirements for their specific security needs including those following the CSfC specifications. Thus, our FPGA prototype does not aim to replace existing systems, but instead offer an additional method for cloud tenants control over their data security.

5.7 Conclusions and Future Work

The prototype system presented in this chapter is a step toward an FPGA approach to the requirements of Commercial Solutions for Classified through dynamic configuration of features and protocol implementations. One advantage to our prototype is the ability to update and improve the encryption and packet filtering mechanics. This provides tenants with the ability to adopt future encryption schemes when patches are applied against newly discovered vulnerabilities both against the current protocols and any targeted attacks on the system running on the attached node.

As this was a simple proof-of-concept prototype, we demonstrate how FPGAs available in the datacenter provide a configurable environment for future network isolation techniques. Using reconfigurable logic enables tenants to have direct control of advanced security features. Future steps toward a full featured network isolation solution would be to enable a full security suite of packet operations and a certified FSS that can be bootstrapped and attested by private organizations to program, configure, and use all the available features a Bump-in-the-Wire FPGA brings to private cloud enclaves. Integrating FPGAs into the current research — of turning public clouds into private cloud enclaves — can help increase the security of running nodes and also enable future developments in provisioning and network isolation of each private enclave from other nodes in the same datacenter environment.

Chapter 6

Hardware Accelerated Multi-Party Computation

Organizations gain substantial benefits from shared data pools, however not all organizations are able to share their private data due to privacy concerns. These organizations cannot use trusted third parties to jointly compute across data in fear of potential data leakage. Thus, much research has been made in the field of privacy preserving computation. However, many of these approaches suffer from high computation requirements and continue to be a limiting factor for adoption.

In this chapter, we argue that FPGAs in the datacenter provide an ideal environment for crytographically secure algorithms. Specifically we look to improve the performance of secret sharing Multi-Party Computation using the hardware features available through network connected FPGAs. We first give background about MPC and datacenter adoption section 6.2. Then we discuss our three party algorithm of choice in subsection 6.2.3 and corresponding FPGA designs section 6.3. Next we look into a honest majority four party algorithm in subsection 6.2.4. We adopt the COPA FPGA environment to design a network connected lookaside accelerator in section 6.4. Finally, we describe a future application of our hardware accelerated MPC for Machine Learning in section 6.5. In this section, we present our current progress towards a complete software-hardware distributed MPC Machine Learning case-study and describe the unique opportunities datacenter MPC provides for future Federated Learning applications.

6.1 Introduction

Multi-party computation (MPC) allows parties to securely share confidential data and compute collective information without ever releasing one's own personal data. As defined in pending legislation within the United States Senate, "the term 'secure multi-party computation' means a computerized system that enables different participating entities in possession of private sets of data to link and aggregate their data sets for the exclusive purpose of performing a finite number of pre-approved computations without transferring or otherwise revealing any private data to each other or anyone else" (Wyden, 2019). As discussed previously chapter 5, cloud systems for confidential data have been limited to privately owned systems and are less widely utilized by those requiring more computation power at any given moment. Thus, the public cloud is rarely considered by those interacting with confidential data.

MPC has been an active area of research for the last 40 years (Evans et al., 2018; Yao, 1982; Yao, 1986; Shamir, 1979), and it has been deployed to protect data in the healthcare (Archer et al., 2018; Giannopoulos and Mouris, 2018), education (Bogdanov et al., 2016; Feigenbaum et al., 2004), finance (Bogetoft et al., 2009; Damgård et al., 2017; Abidin et al., 2016), and technology (Bonawitz et al., 2017; Ion et al., 2019). In addition, the use of MPC for machine learning applications is actively being researched (Bonawitz et al., 2017; Mohassel and Zhang, 2017; Mohassel and Rindal, 2018; Wagh et al., 2020; Dalskov et al., 2021; Byali et al., 2020; Chaudhari et al., 2020). We go into further details for our ongoing implementation efforts toward distributed Machine Learning applications and potential future applications of Federated Learning in section 6.5.

Recent surveys reveal a few companies with specialized MPC offerings. For adoption of MPC to increase, it is necessary to continue to improve the performance and ease of use of general-purpose systems. Existing work shows that general-purpose

MPC can be used in viable systems (Huang et al., 2012).

6.2 Background

MPC uses a combination of 4 operations, known as gates, to perform all computation requirements. The combination of the 4 gates form a Turing complete machine allowing for any algorithmic computation to be performed using MPC. The focus for research in MPC is to improve the performance to these gates in order to improve the overall performance of the application.

MPC currently focuses on two main approaches. Secret Sharing allows for a balanced arithmetic intensity between compute and network bandwidth, but it relies on a low latency environment to achieve high performance results. In contrast, Garbled Circuits typically perform better in higher latency situations, due to fewer communication rounds, but require high bandwidth to support large circuits. Because Garbled Circuit approaches are typically compute-bound, they appear more amenable to hardware acceleration, and as such have been the subject of significant prior research with FPGAs (Järvinen et al., 2010a; Järvinen et al., 2010b; Frederiksen et al., 2014; Songhori et al., 2016; Hussain et al., 2018; Songhori et al., 2019; Hussain and Koushanfar, 2019; Fang et al., 2017; Fang et al., 2019; Huang et al., 2019; Leeser et al., 2019). The current approach to MPC on FPGAs may need to be reconsidered to better utilize a growing and evolving cloud environment with the introduction of FPGA SmartNICs and change to a data-centric datacenter.

6.2.1 Datacenter Model

The target datacenter model used here is the adoption of a MPC-as-a-Service which can utilize the various configurations of accelerators found in different cloud and cluster setups. The MPC use case requires multiple computing and communicating parties for security and low latency networking for performance. Thus we consider

scenarios where datacenter processing hardware is owned by different parties and co-located within a single physical location. This helps to limit the additional bottlenecks found when traversing physical distances to perform low-latency operations.

FPGA hardware acceleration has seen increasing adoption in datacenters due to their very high bandwidth and low-latency communication used by means of high speed transceivers. As described in subsection 6.2.2, FPGA hardware properties, especially co-location of compute and communication logic on the same device, yield high throughput for MPC protocols based on Secret Sharing, which makes the most effective use of available bandwidth.

We believe the availability of network-attached FPGA configurations in the datacenter provide the most beneficial scenario for MPC secret sharing computation. Our initial experiment platform is done using FPGAs readily available on commercial clouds in a co-processor accelerator model. We aim to show a scenario where parties are co-located on a single FPGA to provide an ideal low latency communication environment. Considering the security and confidentiality concerns with co-located parties on a single hardware resource, our successive experiments examine MPC hardware acceleration with FPGAs SmartNICs using the COPA framework.

6.2.2 MPC Paradigms

In general, MPC protocols allow an arbitrary number of compute parties N to perform a joint computation while resisting a subset of T 'bad' parties who wish to breach the confidentiality of other people's data or tamper with the integrity of the calculation. In our first experimental setup, we examine a 3-party protocol which tolerates 1 adversarial party who "semi-honestly" follows the protocol and only tries to break confidentiality. Definition of the protocol can be found in subsection 6.2.3. Our following experiments utilize a 4-party protcol with protection against 1 malicious party. The protocol includes features which allow each party to perform checks after

every independent operation with another party member. Additional definitions for the protocol can be found in subsection 6.2.4.

General-purpose MPC designs often represent the agreed-upon computation as an arithmetic or Boolean circuit, and follow the Garbled Circuit or Secret Sharing approaches. Secret Sharing-based MPC systems have the compute parties evaluate each gate of the circuit in parallel on their own pieces or *shares* of the data, with a small amount of network communication required for each multiplication or AND gate (none is required for addition or XOR gates).

The computation and communication overhead of MPC manifests itself differently for Garbled Circuits and Secret Sharing. Even with optimizations (Beaver et al., 1990; Naor et al., 1999; Kolesnikov and Schneider, 2008; Zahur et al., 2015; Yakoubov, 2017), Garbled Circuits have a small number of communication rounds but a large communication size (80-128× the size of the original data), rendering them beneficial in high-latency scenarios but detrimental when processing large datasets. Conversely, Secret Sharing approaches require a low-latency environment because they involve many rounds of communication, however they consume substantially less bandwidth per computational step.

To date, most MPC implementations are in software, and thus rely on general-purpose processing hardware and commodity networking equipment. In this scenario, Secret Sharing tends to be network latency-bound whereas Garbled Circuits are often compute-bound. Consequently, most of the prior focus in hardware acceleration has been directed toward Garbled Circuits. Our work specifically considers MPC implementations in the datacenter, where Secret Sharing systems offer higher maximum throughput and the network latency can be low enough to realize meaningful performance benefits by optimizing the computation with FPGAs.

6.2.3 Three Party Semi-Honest Secret Sharing MPC

The original Secret Sharing implementation (Araki et al., 2016b; Araki et al., 2016a) differentiates between arithmetic and Boolean operations. The design by Araki et al. requires that 3 parties agree to perform a computation, however it only requires each party to communicate with one other party at a time (aside from initial secret sharing and revealing the final result). This means for most actions 1 adversarial party is tolerated, but breaks the trust in the system when employed in higher application scenarios.

We focus on the implementation of both types of Secret Sharing circuits, Boolean and arithmetic. The main difference between these two categories is based around the use of a ring modulo 2^n. For all $n > 1$ operations are categorized into arithmetic gates either addition or multiplication. We opted to use $n = 128$ for our arithmetic implementation. For the special case $n == 1$, addition is the same as XOR and multiplication is the same as AND.

The process of MPC secret sharing involves 3 phases. First, confidential data is securely split into three *shares* and given to each party member. Each party's share does not contain enough information to deduce the original confidential data. The protocol dictates that only 2 parties' shares are necessary to reconstruct the final result. Computation is performed using either the Boolean XOR and AND gates or arithmetic addition, and multiplication gates. The decision to use either form of Secret Sharing is made as a group between all parties either before the share splitting phase occurs or after a transformation is done between secret sharing protocols. Reconstruction of the final value is done with the help of another party member's final share information. We explore the details of each step below for the protocol implemented.

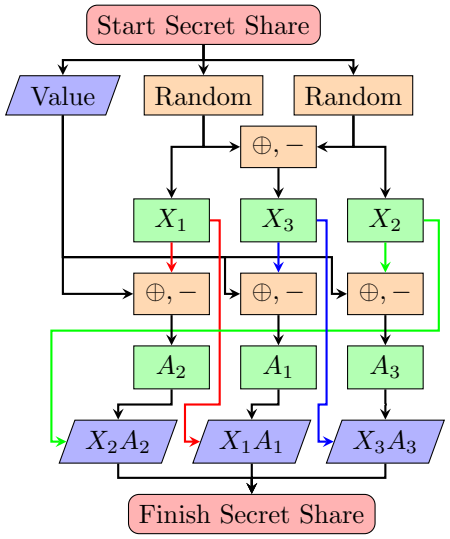


Figure 6·1: Initial construction of three party secret shares. Two pseudo-random values are selected and used to compute three values following Equation 6.2. The second piece is constructed with the random values and each pair is sent around to one party member.

Data Split - Share construction

Each share consists of two pieces: one uniformly random value, and one value based on data passed from another party. We denote each share as a pair (x_i, a_i) and v as our confidential data being shared among the party. Shares are generated and distributed based on the computation being performed, either arithmetic or Boolean logic. Shares are generated using uniformly random values x_1, x_2, x_3 which follow the rule stated in Equation 6.1. Based on the protocol form of Secret Sharing, values for (x_i, a_i) are created using either Equation 6.2 or Equation 6.3, where if $i = 1$ then $i - 1 = 3$. The process of generating three unique shares for the confidential value v is visualized in Figure 6·1.

$$x_1, x_2, x_3 \in \mathbb{Z}_{2^n} \tag{6.1}$$

Arithmetic:

$$x_1 + x_2 + x_3 = 0 \text{ then } a_i = x_{i-1} - v \tag{6.2}$$

Boolean:

$$x_1 \oplus x_2 \oplus x_3 = 0 \text{ then } a_i = x_{i-1} \oplus v \tag{6.3}$$

Gate operations - Computation phase

To perform any operation two shares are passed to each party, one for each data value 'v_1, v_2'. We denote an input pair for v_1 as (x, a), v_2 as (y, b) and the output of the operation as (z, c).

- *XOR* - Each party can compute the XOR of their individual shares simply by performing a local xor of the individual parts in the pair.

$$z = x \oplus y \text{ and } c = a \oplus b \tag{6.4}$$

- *Addition* - Similar to XOR, individual parties can perform a simple addition of arithmetic shares using their local information without the need for communication.

$$z = x + y \text{ and } c = a + b \tag{6.5}$$

- *AND* - Performing an AND operation requires parties to communicate with one another and thus is more complex. First, each party i produces a correlated random value $\alpha_i \in \{0, 1\}$ which follows Equation 6.6. From here we calculate r_i using Equation 6.7. Lastly, each party passes their r_i value to one other party member and finally calculates the output share (z, c).

$$\alpha_1 \oplus \alpha_2 \oplus \alpha_3 = 0 \tag{6.6}$$

$$r = x\&y \oplus a\&b \oplus \alpha \tag{6.7}$$

$$z = r_i \oplus r_{i-1} \text{ and } c = r_i \tag{6.8}$$

- *Multiply* - Much of the computation for multiply is similar to the AND gate

previously described. Parties select correlated random values that hold $\alpha_i \in \mathbb{Z}_{2^n}$, following Equation 6.9. Calculation of each r_i is done using Equation 6.10. Due to the unique nature of the shares and their set associativity, we take advantage of the modular multiplicative inverse for our q value. Similar to the communication necessary in the AND gate, each party passes their generated r_i and calculates their individual shares (z, c).

$$\alpha_1 + \alpha_2 + \alpha_3 = 0 \tag{6.9}$$

$$r = (a \cdot b - x \cdot y + \alpha) \cdot q \text{ where } q \cdot 3 \equiv 1 \pmod{2^n} \tag{6.10}$$

$$z = r_{i-1} - r_i \text{ and } c = -2r_{i-1} - r_i \tag{6.11}$$

Final results - Data reconstruction

Finally once all computation is performed on the party shares, each member can reconstruct the final value by requesting information from one other party member.
Arithmetic:

$$v' = z_{i-1} - c_i \tag{6.12}$$

Boolean:

$$v' = z_{i-1} \oplus c_i. \tag{6.13}$$

6.2.4 Four Party Honest Majority Secret Sharing MPC

We use the 4-party honest-majority MPC protocol of Dalskov, et al. (Dalskov et al., 2021) with malicious security. This design requires 4 parties to agree on joint computation in a 3-out-of-4 protocol. This means a single party member maintains just enough information to only require communication with one other member to reconstruct the final computation. Similar to the previous 3-party protocol, operations are split between arithmetic and Boolean depending on the ring modulo 2^n.

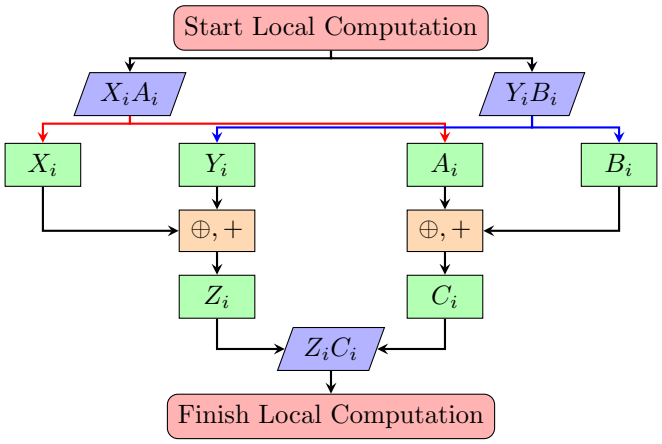


Figure 6·2: Local computation of XOR, addition gate

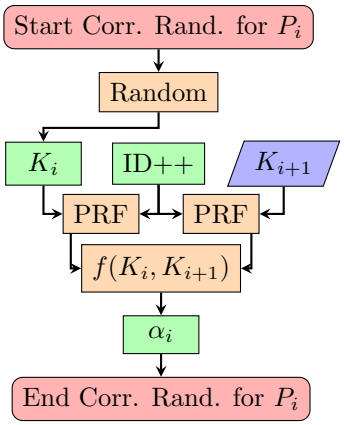


Figure 6·3: Three party correlated random values are generated with a combination of a local key and one shared key from another party member.

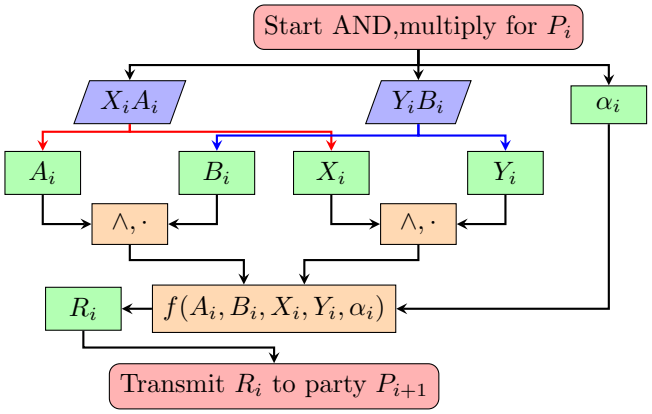


Figure 6·4: Three party initial AND, and multiply computation. Intermediate data is exchanged with one other party.

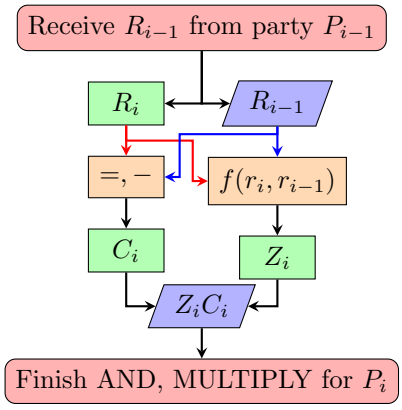


Figure 6·5: Three party final computation performed after communication from one other party member.

Data Split - Share construction

Our initial experiment setup assumes each party member has control of a single FPGA SmartNIC in a Bump-in-the-Wire scenario tied to a host process under their control. Utilizing this hardware setup, we envision that all confidential data is in possession of the node and the FPGA exclusively interacts with only data in share format. As all confidential data remains tied to the host, this maintains full privacy of confidential data as any data passing through the COPA network is obfuscated in the form of shares.

Shares are generated into four pieces: three uniformly random values, and one value calculated with the input data. We denote each share as x_i and v as our confidential data being shared among the party. Shares are generated and distributed based on the computation being performed, either arithmetic or Boolean logic. Shares are generated using uniformly random values x_1, x_2, x_3 which follow the rule stated in Equation 6.14. Based on the protocol form, values for x_i are created using either Equation 6.15 or Equation 6.16. The process of generating four unique shares for the confidential value v is visualized in Figure 6·6.

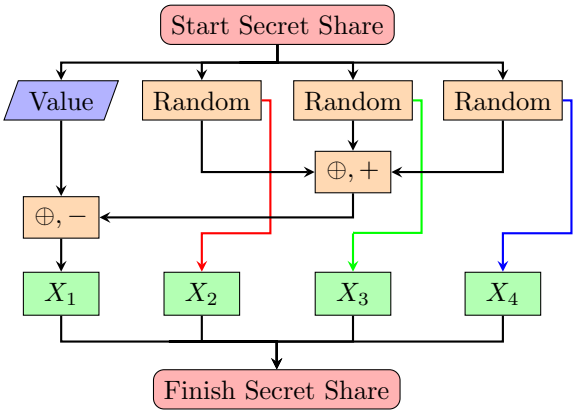


Figure 6·6: Initial construction of four party secret shares. Three pseudo-random values are selected and used to compute the final share value. Values are sent around to party members to follow the 3-out-of-4 share distribution.

$$x_1, x_2, x_3, x_4 \in \mathbb{Z}_{2^n} \tag{6.14}$$

Arithmetic:

$$x_1 + x_2 + x_3 + x_4 = v \tag{6.15}$$

Boolean:

$$x_1 \oplus x_2 \oplus x_3 \oplus x_4 = v \tag{6.16}$$

Gate Operations - Computation phase

We focus on arithmetic MPC operations which consist of addition and multiplication. Shares are formed using *128-bit* values and all operations are based around modular arithmetic with a ring module of 2^{128}. Functionality for Boolean shares is similar to arithmetic and our implementation allows for these additional operations with a simple selector input to change gate function.

To perform any operation two shares are passed to each party, one for each data value 'v_1, v_2'. We denote an input for v_1 as x, v_2 as y and the output of the operation as z. For our algorithm of choice, MPC addition consists of local computation only

and doesn't require interaction between members. Shown in Equation 6.17, shares can be added in parallel and only require local computation done synchronously between all party members.

Addition:

$$\begin{aligned} z_1 &= x_1 + y_1 \\ z_2 &= x_2 + y_2 \\ z_3 &= x_3 + y_3 \\ z_4 &= x_4 + y_4 \end{aligned} \tag{6.17}$$

The base multiplication algorithm requires both local computation of shares and a single round of communication between party members; we thus focus on the performance obtained through improvement of these communication rounds. Each multiplication operation is broken down into the accumulation of share multiplications as seen in Equation 6.18. Since each party member only contains 3-out-of-4 share information, multiplication cannot be done locally due to the missing share information. There are two scenarios in which the distribution of share information creates unique situations during multiplication operation.

Multiplication:

$$\begin{aligned} x * y &= x_1y_1 + x_1y_2 + x_1y_3 + x_1y_4 \\ &= x_2y_1 + x_2y_2 + x_2y_3 + x_2y_4 \\ &= x_3y_1 + x_3y_2 + x_3y_3 + x_3y_4 \\ &= x_4y_1 + x_4y_2 + x_4y_3 + x_4y_4 \end{aligned} \tag{6.18}$$

The first case occurs when performing the individual share multiplication of two of the same shares x_i and y_i, shown in Table 6.1. Three of the four party members contain enough information to perform the local computation. In this scenario, no communication occurs as the party members which need the result z_i' contain all the necessary information. Local computation is performed and is included in the final

Table 6.1: Party '3-1' missing information

Share	Party 1	Party 2	Party 3	Party 4
z_1'		x_1y_1	x_1y_1	x_1y_1
z_2'	x_2y_2		x_2y_2	x_2y_2
z_3'	x_3y_3	x_3y_3		x_3y_3
z_4'	x_4y_4	x_4y_4	x_4y_4	

Table 6.2: Party '2-2' missing information

Share	Party 1	Party 2	Party 3	Party 4
z_1'', z_2''			$x_1y_2 + x_2y_1$	$x_1y_2 + x_2y_1$
z_1'', z_3''		$x_1y_3 + x_3y_1$		$x_1y_3 + x_3y_1$
z_1'', z_4''		$x_1y_4 + x_4y_1$	$x_1y_4 + x_4y_1$	
z_2'', z_3''	$x_2y_3 + x_3y_2$			$x_2y_3 + x_3y_2$
z_2'', z_4''	$x_2y_4 + x_4y_2$		$x_2y_4 + x_4y_2$	
z_3'', z_4''	$x_3y_4 + x_4y_3$	$x_3y_4 + x_4y_3$		

multiplication result.

The second case is unique due to party members missing exactly half of their required information to fully complete the computation. As we demonstrate in Table 6.2, only two party members contain enough information to perform the complete computation and the remaining two party members only have half of the necessary information. We will refer to this step as the '2-2' computation. To perform an accurate and secure '2-2' computation, the process involves usage of a correlated random value between the knowledgeable party members and one communication round with the remaining members.

Correlated random values are generated in the same fashion as done in Figure 6·3, each party member maintains hold of a shared key with one other party member. The '2-2' calculation involves exactly two computations between a pair of party members, therefore six random numbers will be generated two for each shared key.

Parties with full knowledge of sub-calculations in '2-2' perform the local compu-

tation with their information. Directly sharing this information with the other two parties would result in a convergence in knowledge and thus reduce the security of the operation. Instead the correlated random number is used to create a new sharing pair with the completed local computation shown in Figure 6·7. These values will be shared with the remaining party members to complete the operation.

Prior to the communication round, each party member combines the earlier computed $x_i * y_i$ result and two selected splits from each pair of the local computations in '2-2' per share. This pre-communication requirement is shown in Figure 6·8. They store this value in local memory awaiting the values shared by the other party members to fully complete the multiplication operation.

Communication phase

Each pair of party members perform individual '2-2' and jointly choose which piece should be sent to the remaining parties. For base protocol functionality, only a single value must be obtained from each other party member to have an accurate and complete multiplication operation. Since the '2-2' computation involves two party members jointly computing the same information a unique opportunity is created to add additional malicious security. Malicious security is done with the second party member sharing additional information to the same destination as the first. The second round of communication typically involves creating a hashed value of completed share information and performing another round of communication upon completion of the above steps. Hashes of reduced size are typically used to minimize the total amount of communication bits in the overall process, which reduces the overall bandwidth requirement of the added malicious security step. The locally generated hashed value and transmitted value over the network are compared and a mismatch in calculated hashes would indicate computation has either failed or altered by a party member. The party members can then abort future computation to avoid

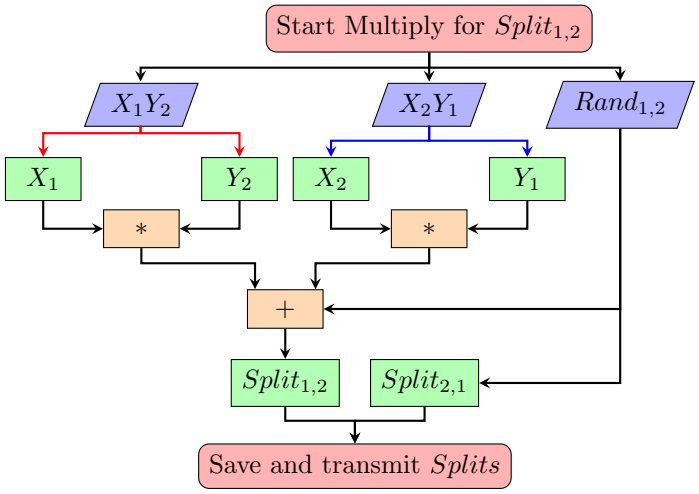


Figure 6·7: Each '2-2' computation generates a unique pair of split information determined by the local computation performed by the knowledgeable party members and a correlated random value. This pair provides no additional knowledge into the initial share information when used during the communication stage and thus helps to maintain privacy of the operation.

the risk of data leakage in the case of a malicious actor. Since all intermediate data is in the form of shares, no additional information would be available by a single party member, and trusted members will not share any data after aborting the current process.

6.3 FPGA accelerated Three Party Secret Sharing MPC

We focus on the Araki et al. Secret Sharing using 3 parties as our starting point. Extensions of this protocol can provide more security over malicious individuals or allow for more party members(Araki et al., 2017; Furukawa et al., 2017; Furukawa and Lindell, 2019). Our goal is to provide evidence toward the benefits of accelerating MPC in the datacenter while leaving room to adapt the protocol for future use cases.

Our designs for the arithmetic and Boolean versions of Secret Sharing contain many common details and features. This provides the opportunity to customize the allocation and utilization of gates in the design at a higher level without the need to

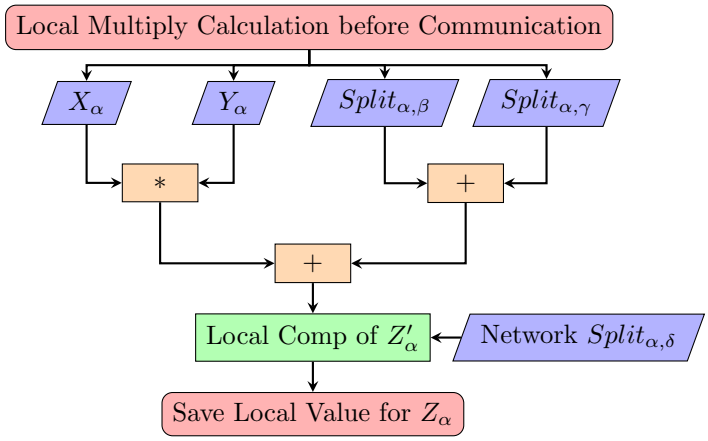


Figure 6·8: Each party member perform all local computation for both '3-1' and '2-2' split pair generation. The local information is combined together and waits for network communication from a single party member to complete the multiplication operation.

reconfigure the hardware. We will discuss in depth the process involved when utilizing *AND* and *multiply* gates as those require different calculation and communication actions. XOR and addition gates are done locally and do not require much additional design choices as seen in Figure 6·2.

We rely on OpenCores projects (Hsing, 2012; Castillo, 2004) for our design when generating pseudo-random numbers. This design choice was made out of convenience to easily adapt the design to any hardware provided. Our future plans are to utilize vendor specific Random Number Generator (RNG) hard cores or reference designs created by FPGA vendors to provide a better optimized version for targeted hardware. Security of the OpenCores RNG module was not considered for this application but will be examined in the future when making a selection between different vendor specific tools and designs. We use the selected random number during initial key generation, share splitting phase, and the correlated random requirements in both AND and multiply gates.

The usage of a correlated random number between each of the three parties is important to both following the protocol listed in section 6.2.3 as well as providing

a method for all parties to obtain random values without the need for additional communication. Figure 6·3 shows the process we developed for our correlated random protocol. Prior to performing any MPC computations each party generates an initial key through the use of the RNG block and maintains a copy of the key value and shares this same value with one other party member in round robin fashion. These keys are then used to generate a pseudo-random value of 128-bits using both keys in possession. The function $f(K_i, K_{i+1})$ is pre-determined when the party selects to use Boolean or arithmetic operations.

Arithmetic:

$$\alpha_1 + \alpha_2 + \alpha_3 = 0 \text{ and } \alpha_i = PRF(K_i) - PRF(K_{i+1}) \tag{6.19}$$

Boolean:

$$\alpha_1 \oplus \alpha_2 \oplus \alpha_3 = 0 \text{ and } \alpha_i = PRF(K_i) \oplus PRF(K_{i+1}) \tag{6.20}$$

Calculation of α must be performed for each use of an *AND* or *multiply* gate to provide additional security to the protocol and prevent replay attacks on the system.

6.3.1 Analysis of FPGA Implementation

The FPGA implementation uses Amazon Web Services (AWS) FPGAs available through its Elastic Compute Cloud (Amazon EC2). Specifically, Xilinx Virtex UltraScale+ VU9P FPGAs are accessible via a virtual machine in EC2 F1 instances. Amazon includes a FPGA hardware *shell* for software/hardware co-design between the node CPU (Intel Xeon E5-2686 v4) and FPGA. Software control for F1 instances rely on provided DMA functions and PCIe function templates which communicate directly with the hardware shell (Amazon, 2016). This furnishes the mechanism for loading data, controlling operations, and retrieving results.

The PCIe packets are translated through the Amazon shell and utilize multiple AXI bus configurations to send and receive data with the software system. We use

the general purpose AXI bus supporting a 512 bit data packet to provide a single message containing two secret share vectors (4×128-bits) prior to starting the hardware operations. The HDL design takes each AXI bus message, parses the information, and relays data to the desired module.

Since all gates are routed individually, connecting gates to form a specific circuit or equation can be done either through the software host process or with an agreed upon hardware change between all party members. This provides us the opportunity to route multiply and addition gates to function together for matrix multiply as individual connections or as a joint multiply accumulate (MAC) module. Previous work done with matrix multiply shows efficient methods of using Garbled Circuits to accelerate the generation of MPC circuitry before passing the garble tables over to the participating party member (Hussain et al., 2018; Huang et al., 2019; Leeser et al., 2019). Our approach to matrix multiply using secret sharing performs all computation jointly during run-time. We discuss later in subsection 6.3.3 how our FPGA implementation can fully utilize the network bandwidth found in the datacenter for larger applications such as machine learning.

6.3.2 Results

MPC Secret Sharing modules are assessed in terms of total FPGA resource utilization and total throughput with all gates running in parallel.

Our final implementation requires only a single cycle to perform either an XOR or addition gate and a total of 4 cycles to calculate an AND or multiply gate. This is obtainable after an initial 21 cycles of pre-computation of party keys and initial correlated random values. The computation for each additional correlated random value is performed during the utilization of the current peusdo-random value during AND or multiply gate evaluation.

At the time of our initial three party experiments, Amazon F1 did not currently

offer their "FPGA-Link" and thus only provides FPGA to FPGA connections through PCIe or over their 25GbE network through a host NIC. Our current design focuses on obtaining the most utilization on a single FPGA and thus is not limited based on network speed. For verification and to fully utilize our design without network bottlenecks, we designed a test environment allowing for 3 parties on a single FPGA. This is a possible use case in a real world environment through the use of a trusted third party to perform calculations on behalf of all three party members. This design includes the necessary routing and control logic to enable the system to perform calculations between all parties from start to end without the need of software intervention. The test environment allocates 3 AND or multiply gates, one for each party member, then each group is duplicated to fully utilize the FPGA; summarized in Table I."

We include the number of bits in Table 6.3 as a measure of the amount of data communicated during the exchange part of each computation cycle.

Amazon F1 instances provide a few choices in clock speed when synthesizing FPGA designs. As a conservative estimate the default choice of 125Mhz is selected as we are not currently limited by a computation bottleneck. We determine that the current design utilizing 4 cycles per AND operation and running at 125Mhz has the potential to perform 160 Gbps of computations when being fully utilized with only 20% utilization exceeding the current Amazon F1 network bandwidth.

Comparing with the Araki et al. design running on 20 Xeon E5-2686 v4 cores, their theoretical MPC op./sec can reach a total of 10 billion operations, while our fully utilized FPGA design surpasses it by over 10×.

6.3.3 Analysis

Based on the minimal data dependencies and flow in Figure 6·3, Figure 6·4, and Figure 6·3, in principle all MPC Secret Sharing gates can execute one operation per

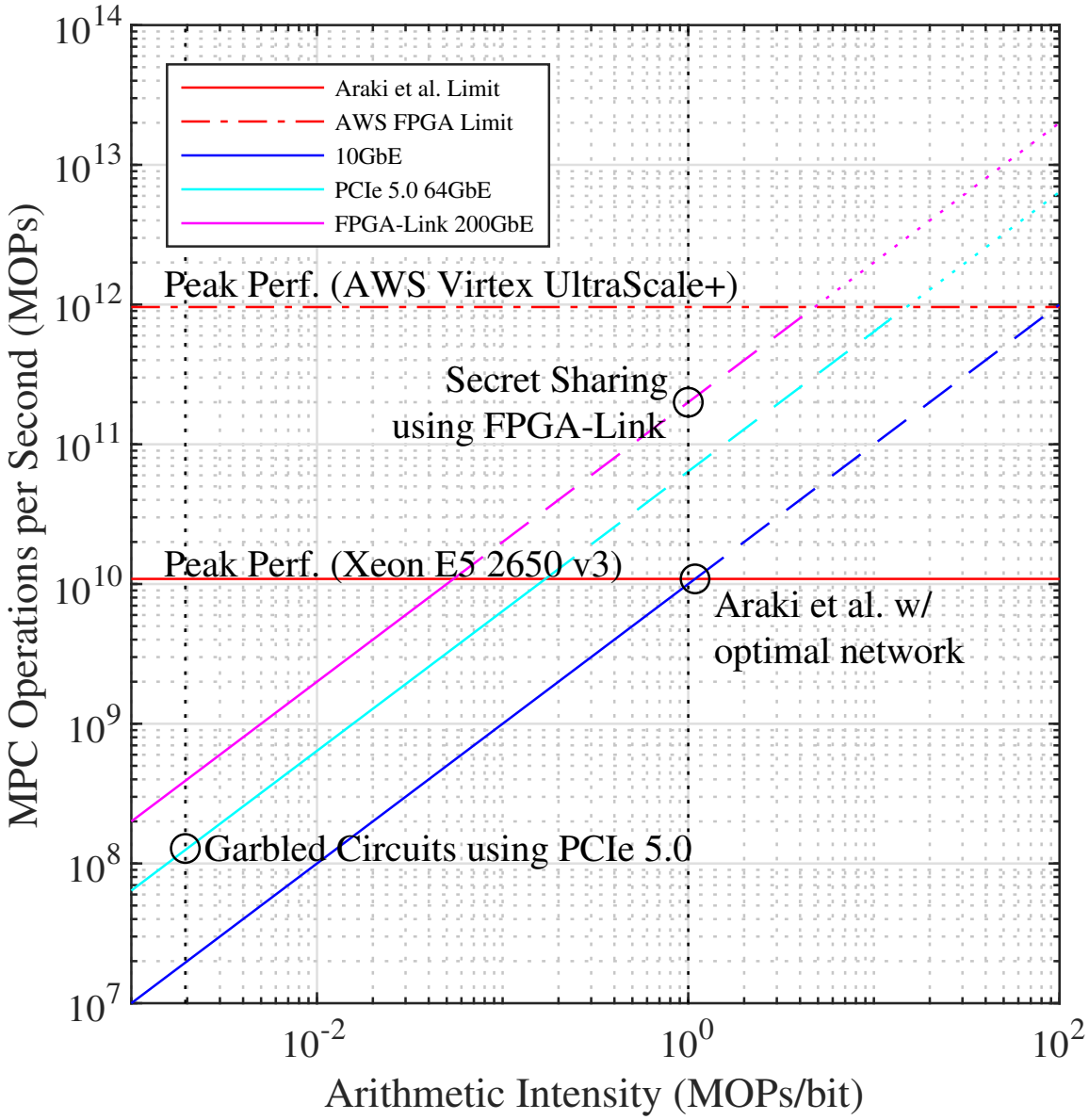


Figure 6·9: Roofline model comparing Secret Sharing performance against datacenter bandwidth. We denote limitations in the Araki et al. design and our FPGA Secret Sharing implementation, and FPGA implementations based on Garbled Circuits.

Table 6.3: AWS Implementation Result Analysis

AND Cores	Bits	MPC (billions op.)/sec
1	128	2.67
3	384	8.00
12	1536	32.0
24	3072	64.0
48	6144	128
60	7680	160

clock cycle. This can be done by utilizing a fully pipelined implementation of the AND and multiply gate. Our follow-up work shows the possibility of a fully pipelined one operation per clock cycle design in section 6.4. Such a design would saturate a 10Gbps network connection when operated at 78.13 MHz. Operating at the higher frequencies used commonly by FPGAs would require higher bandwidth.

Our tests and analysis of different quantities of MPC blocks on Amazon AWS provide sufficient data points to establish that Secret Sharing MPC can be competitive when implemented on FPGA hardware in the datacenter.

The original software design utilized a software implementation of the protocol executed on general purpose processors (Araki et al., 2016b; Araki et al., 2016a). Specifically, each party used varying numbers of cores from one or two Xeon processors. The authors were able to nearly saturate their 10Gbps link (7.38Gbps) between parties when using all cores in each node. While the authors were limited by multiple cores causing queuing congestion at the Network Interface Card (NIC) the use of a CPU appears to have more limited scaling potential. Using the reported number of AES/sec and network communication for 1 core, scaling from 73.3% CPU usage to 100% would appear to show 1 core being capable of saturating a $\sim$0.780Gbps connection. Multiplying for 20 cores that would indicate a peak of $\sim$15.6Gbps.

In comparison, the FPGA block we tested for performing Secret Sharing only

requires 3 AND cores to exceed the 7.38Gbps reached with 20 CPU cores, reaching a full 8.00Gbps. This uses ∼5% of the fabric available on the FPGA targeted, a 10× improvement vs the CPU utilization. Attempting to fully employ the available fabric it is possible to implement 60 AND cores based on our design which would permit saturation of a 160Gbps link.

We demonstrate the peak performance obtainable using the roofline model in Figure 6·9. Figure 6·9 shows the limiting factors and constraints of Secret Sharing designs: compute bandwidth (horizontal lines) and network bandwidth (diagonal lines). We first show that the peak performance is demonstrated in number of MPC operations per second (MOPs), while arithmetic intensity (AI) is designated as MOPs/bit of data transferred over the network. We utilize the AND gate as our common Secret Sharing gate and assume a pipelined design of one MOP per cycle. Each Secret Sharing gate requires exactly one bit of communication thus an arithmetic intensity of 1 is achievable. Therefore the Araki et al. design has the potential to hit a peak performance of 10^{10} MOPs if they did not encounter any drawback due to network congestion. Comparatively we show that a fully utilized FPGA running all AND gates can outperform the software design by ∼2 orders of magnitude. This analysis shows that the Araki et al system provides an almost perfect balance between network and MOPs when utilizing their 10Gbps interface. However, if a larger bandwidth network is available, such as 40 GbE, then their application will be limited by computational performance and thus will not effectively utilize the bandwidth available.

By contrast, using a fully pipelined version of our design, we show a theoretical peak performance of the FPGA accelerated Secret Sharing design able to achieve 10^{12} MOPs. This indicates that the design of Secret Sharing on FPGAs remains limited by the current bandwidth offered in the datacenter. This limitation allows for the extra FPGA resources to be distributed for other tasks such as more local computations

of *XOR* and *addition* gates or control logic for high performance applications. We compare this against the arithmetic intensity of Garbled Circuits with the assumption that each garbling table can be created each clock cycle. Traditionally each MPC gate contains a table of 4 SHA hashes. We consider the arithmetic intensity (AI) of Garbled Circuits to be $1/(4 \times 128) = 0.002$, which when translated to the roofline model shows that Garbled Circuits are heavily network limited due to the low arithmetic intensity and high bandwidth requirements.

These results demonstrate preferable scaling properties supporting the selection of FPGAs for acceleration. Based on the results, targeting the anticipated 200Gbps links in the Amazon F1 would require less than 25% fabric utilization to reach full saturation utilizing the pipeline improvement described earlier. With a frequency improvement, adjustable through AWS, even less fabric would be required. The remaining available fabric is beneficial as it allows for work distribution, additional secure computations, and higher application control logic such as matrix multiply. In the follow-up section we look into our design of a 4-party computation protocol and achieve high performance through the use of an FPGA SmartNIC.

6.4 FPGA Four Party Secret Sharing MPC using COPA

In our followup experiments, we utilize the 4-party protocol described in subsection 6.2.4. We adopt the protocol onto the COPA lookaside accelerator for optimal utilization of FPGA compute resources and the 100Gbps COPA network.

To fully utilize the COPA lookaside accelerator we split up the multiplication operations into the local computation stage and a post communication accumulation stage as seen in Figure 6·10. Data is first obtained by the DMA logic and stored into two sets of on-chip memory, Data A and Data B. Data in these two on-chip memory regions are based on the stage of computation being performed. First stage

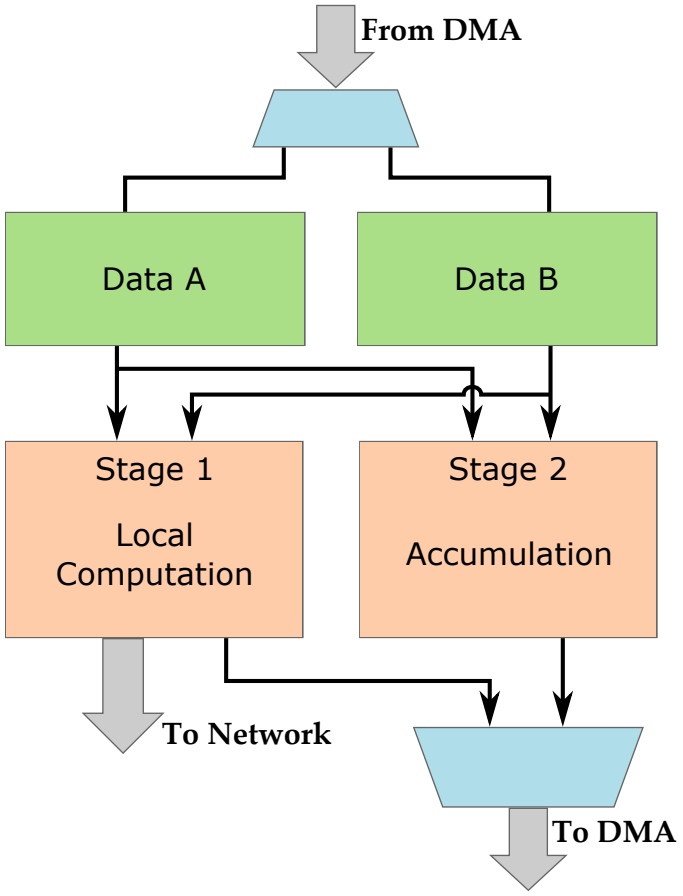


Figure 6·10: Two-stage MPC implementation as single lookaside accelerator. Stage 1 performs local arithmetic operations on incoming share data. Some output is sent back to host memory and other sent out COPA network to computing parties. Stage 2 performs accumulation of local share data and incoming data from parties.

computation takes two lists of values in share form, while the second stage contains the intermediate share data and communication data received from other party members. The local computation stage uses a pseudo random number generator and on-chip resources to perform the initial MPC calculations and save the intermediate share information back to host memory. Addition operations only require the use of the first stage accelerator to perform calculation on input data and generate complete shares.

Table 6.4: Single MPC lookaside accelerator FPGA utilization

Stratix 10 FPGA	Raw (Total Percentage)
Freq	250MHz - 275MHz
ALM	10667 (1%)
Memory bits	5,156,500 (2%)
RAM blocks	668 (6%)
DSP blocks	150 (3%)

Multiplication operations use the first stage to prepare partial local shares and data for communication to other party members. Stage 1 hardware implementation is fully pipelined with the help of parallel random number generators working in unison to produce the six necessary random values per cycle of input data. This results in local computation being available every cycle after initial startup delay.

On completion of the first stage, party members prepare data for the communication phase to pass along via the COPA network. The lookaside accelerator uses dedicate hardware command queues to create PUT commands which simultaneously transmit the prepared data to each correct party member and signal the host system for stage 2 operation on data transmission completion. This process both prepares the data for processing in the second stage of the multiply operation and enables the host to trigger second stage operation. Stage 2 performs a simple accumulation of the locally generated intermediate shares and ingress data from party members, saving the result back into host memory for future computation. In subsection 6.4.2 we will discuss the limitations of the current design and address improvements, this includes the use of remote invocation of the lookaside accelerator and a better coupling of intermediate data between stage 1 and stage 2.

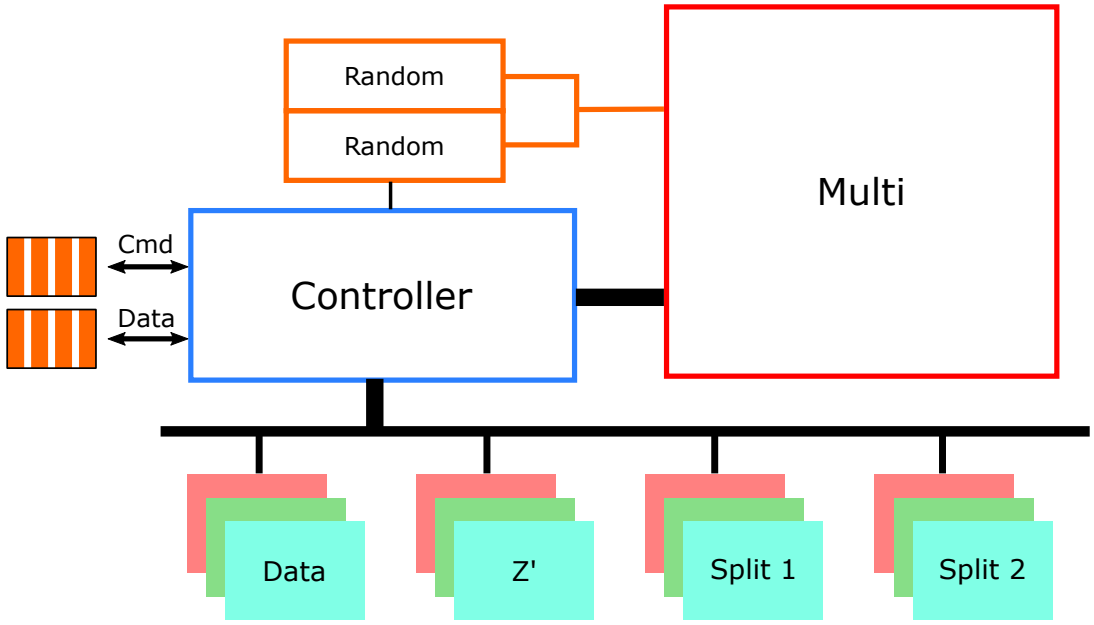


Figure 6·11: Fully pipelined implementation of first stage MPC lookaside accelerator. Commands are sent to the controller which interpret the operation, share structure, and location of data in memory. DMA operations are performed and data is sent to multiplication instance. Parallel random number generators enable new values to be used on every successive clock cycle. Upon completion of the first stage operations, the local output calculation z', and two split pairs are generated and saved in local memory for communication and output DMA operations back to host memory.

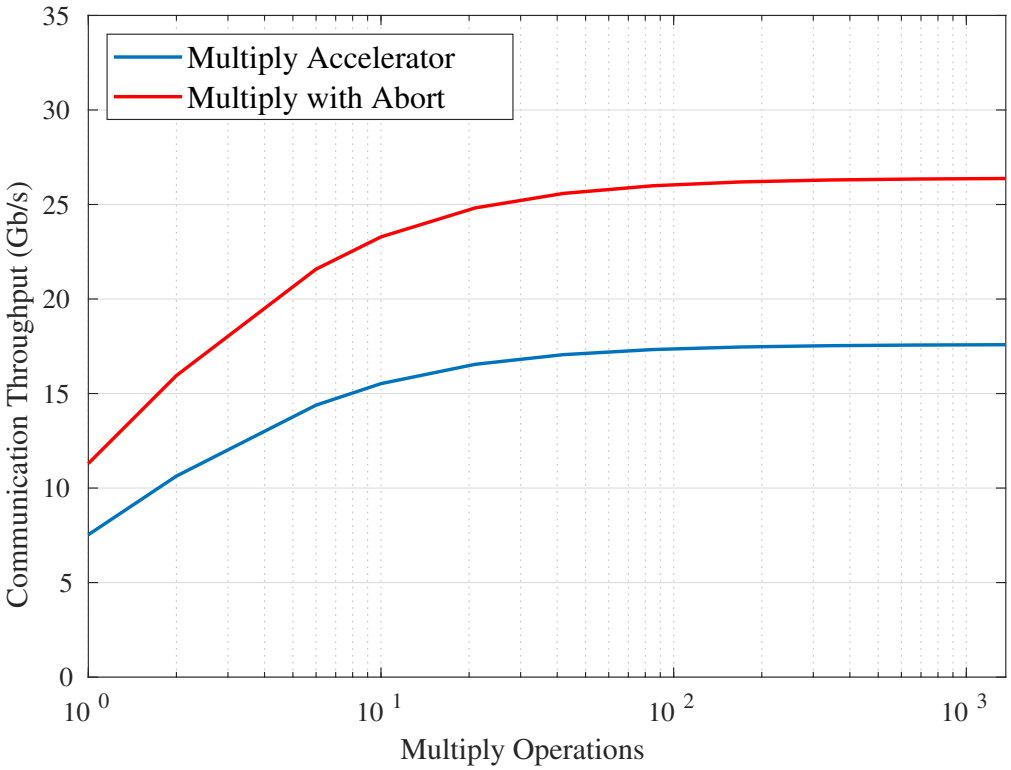


Figure 6·12: Throughput comparison between base MPC accelerator and MPC with malicious security running at 275 MHz with varying batch sizes of multiply operations. Malicious security requires an additional collision resistant hash value to be transferred. Saturation of the base accelerator is over 17.5Gb/s and with malicious security is over 26.3Gb/s.

6.4.1 Experiment Setup

We focus on the throughput and resource utilization of the multiply operation for a 4-party semi-honest majority MPC algorithm (Dalskov et al., 2021). In particular, performing multiplication on shares of *128-bit* integer data types which generates three *128-bit* integers per multiply operation for communication to the other party members. We implement our hardware design on the COPA framework using Intel Stratix 10 FPGAs interconnected with 100GigE high speed transceivers. Each party maintains ownership of a single FPGA connected to a host system using the COPA framework for communication between party members. Acceleration is performed through the use of a single lookaside accelerator, on each FPGA, with dedicated software commands sent from each host system directly to the FPGA through a co-processor acceleration queue. The lookaside command format allows for batch operations on a stream of data from a specified source and saves local computation back to host memory while preparing the network data for transfer to each party member.

In our results we consider both the base algorithm performance and that of included hashed values for malicious security protection. The process of malicious security protection can be disregarded if all parties are semi-honest and will not attempt to stray from correct operation. This is normal for a secure joint computation environment where each party member has more to gain from successful completion of computation then from trying to obtain information about a single party member's confidential data set.

6.4.2 Results & Analysis

Resource utilization for a single MPC lookaside accelerator can be found in Table 6.4. This shows the implementation uses minimal resources which allows for the inclusion

of more accelerators into each COPA FPGA; these additional accelerators may include multiple instances of the MPC core operations or additional functionality for High Performance Computing applications such as collectives similar to the discussion in subsection 4.3.4. With the inclusion of a single MPC accelerator, Figure 6·12 shows how much data is available for communication based on the input length of the lookaside accelerator source data.

The pipeline implementation of the accelerator allows for data to be processed and ready for communication every cycle, after an initial startup delay accessing host memory. Using a single accelerator and batching multiplication operations over a stream of source data, the accelerator performs enough computation to saturate a traditional 10Gb/s link. These results are similar to past implementations (Wolfe et al., 2020; Patel et al., 2020) and show that integration with the COPA system is beneficial to improve the total throughput possible with these hardware implemented MPC operations.

Examining the throughput of large batches of multiplication operations, Figure 6·12 shows a single accelerator performing the basic algorithm (without abort) can saturate a 17.5Gb/s connection, while the inclusion of additional malicious security for abort requires larger than 26.3Gb/s connection to avoid saturation. We can therefore include up to 6 MPC accelerators without abort, or 4 MPC accelerators with abort, to fully utilize the network bandwidth capabilities of the COPA network.

In addition to the communication improvements, the COPA system enables a set-and-forget method for acceleration and communication which frees up each host processor to perform additional non-MPC functions. Queuing operations into the lookaside accelerator, with knowledge that data will be shared appropriately, allows for final completion of each operation without the need to block the process on each step.

Knowledge of future COPA FPGA SmartNIC enhancements can help to make the final use of the MPC lookaside accelerator architecture more friendly to individual tenants of a datacenter working together for a secure joint computation. A combination with secure private cloud enclaves and the accelerator capabilities of the FPGA SmartNIC, we envision a managed system which creates temporary virtual networks (VLAN) to perform joint computation between enclaves. The inclusion of remote invocation would enable individual hosts to perform the desired computation with all tenants without the additional engineering work required to synchronize all tenants FPGA accelerators. In addition, inline encryption acceleration would allow for any data during communication phases to only be accessible by a single destination node and would provide a safeguard to simple man-in-the-middle attacks.

6.5 MPC for Machine Learning

Machine Learning algorithms and models can achieve amazing performance for predictive operations when combined with a large amount of source data. To achieve improved ML models, private organizations employ MPC during both the learning and inference phases. These shared models often provide more accuracy for inference as each organization gets the benefit of private data outside their own. Using MPC allows organization's to develop a shared model without disclosing each organizations private data.

In this section, we examine the potential improvements of FPGA accelerated MPC for Machine Learning and our ongoing process of implementation. We also discuss the unique benefits MPC provides for the special Machine Learning technique of Federated Learning and how our future approach differs from those using techniques such as Fully Homomorphic Encryption.

6.5.1 FPGA Accelerated MPC for Machine Learning

The use of FPGA accelerated MPC requires many implementation hurdles to create a fully functioning Machine Learning training and inference package. We envision tackling these requirements through a bottom-up approach. Earlier in section 6.3 and section 6.4 we provided hardware designs and results of running the essential MPC functions for a fully functioning MPC accelerator. Using this hardware we aim to bridge the software and hardware communication requirements. There are many software libraries (Keller, 2020; Knott et al., 2020; Songhori et al., 2015) which offer high level functionality for machine learning applications using low level MPC. Our future goal is to provide a communication wrapper to allow for any MPC software package access to our FPGA accelerated hardware with minimal changes. We would replace the following machine learning functions with versions compatible with our FPGA hardware running MPC.

- Convolution linear layers
- ReLU activation function
- Matrix Multiply

In Figure 6·13 we demonstrate the throughput obtained by a single accelerator core running directly with our simple software layer. As seen in section 6.3 and section 6.4 each MPC gate utilizes a minimum number of resources, under 3% of FPGA fabric, thus allowing for multiple accelerators running in parallel on a single FPGA. This allows for the highly parallel machine learning functions such as matrix multiply and convolution to run over multiple accelerators.

Our initial results for matrix multiply show the total time for all multiplications varying the size of an $N * N$ matrix. The initial dip is reflective of the throughput of a single accelerator and time for data copy. The performance of a single accelerator is

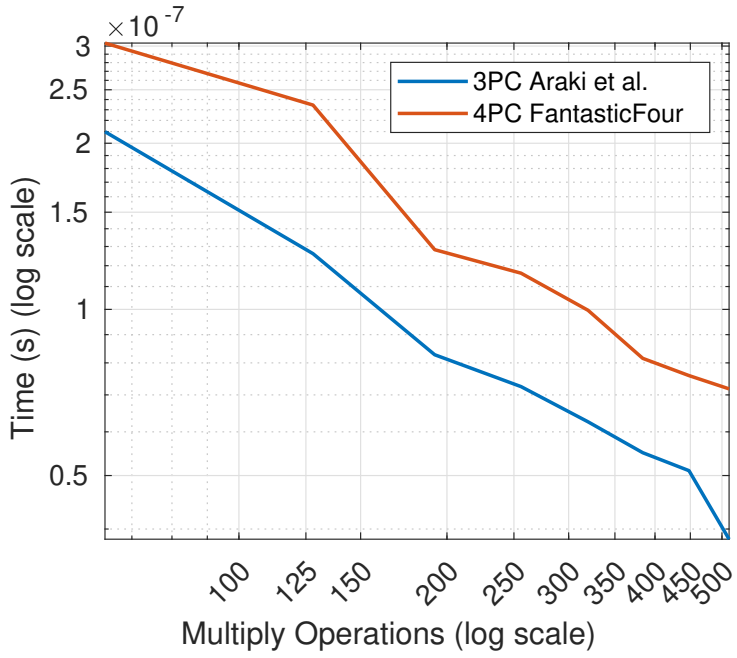


Figure 6·13: Throughput of single multiply gate for machine learning design

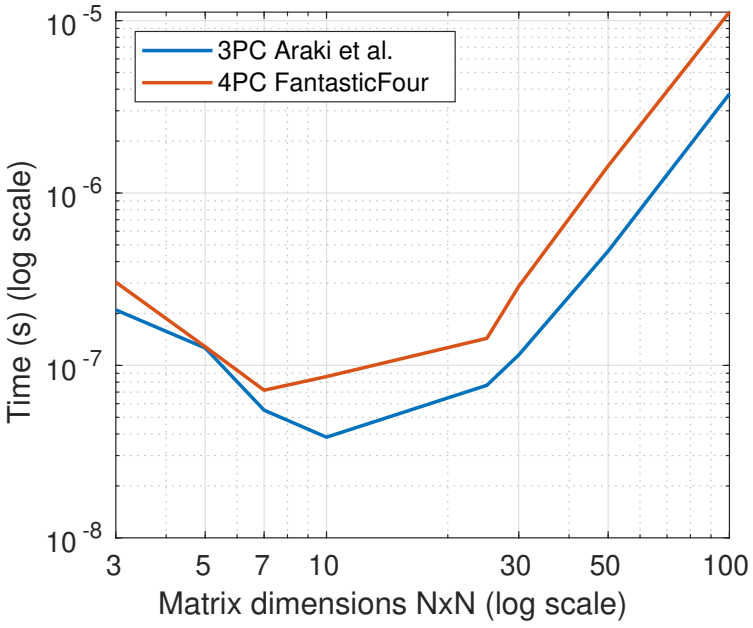


Figure 6·14: Throughput of Matrix Multiply

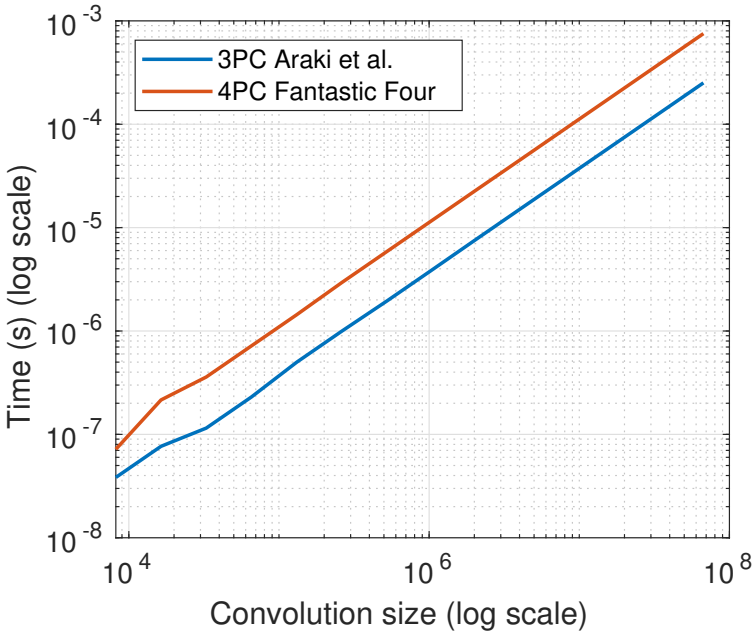


Figure 6·15: Throughput of Convolution

outweighed by the total time for data transfer from host memory to on-board FPGA memory. This is reversed once requirements shift from a single accelerator to multiple accelerators. Here the data transfer is done with a single batch command to all on-board memory and each accelerator operates on their shared memory in parallel. The results for the convolution experiments are based on the total number of parallel multiplication operations. Results differ between the two protocol implementations mainly due to the total amount of data required during data copy and subsequently during each operation communication phase. For our 3PC implementation, each gate requires 4 data points each *128-bit* in size and a single *128-bit* value of communication per party. Comparatively, the 4PC implementation requires 6 data points each *128-bit* and communication of 3 *128-bit* values per party member.

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{6.21}$$

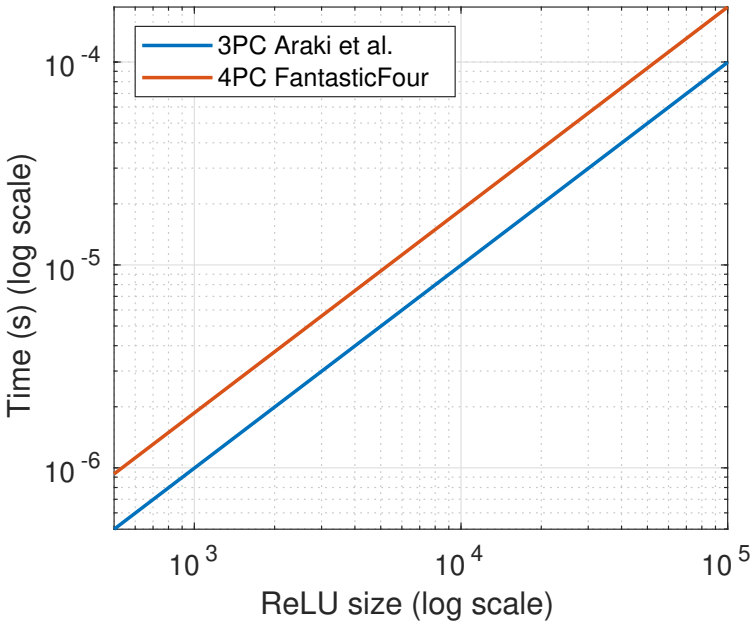


Figure 6·16: Throughput of ReLU

The ReLU activation function requires the evaluation of a piecewise function shown in Equation 6.21. To perform this function in MPC a sequence of events are required.

1. Arithmetic to Boolean conversion of x
2. Obtain MSB to determine sign of value
3. Perform local computation and open value between members
4. Perform arithmetic xor operation
5. Set value to zero if negative

Our initial results are determined based on a total sequence of 13 multiplication operations. We approximate the total time requirements without the inclusion of network latency as our best-case scenario.

We continue to develop the necessary functions for full featured distributed ML training using FPGA hardware assisted MPC. Our eventual goal is to have a comparison between software implementations and a recently published GPU approach using Secret Sharing MPC (Watson et al., 2022).

6.5.2 MPC for Federated Learning

Federated learning aims to enable shared machine learning training between private data owners without the risk of data leakage. The basic method can be condensed into 3 steps. In the first step all data owners start with a global model obtained from a trusted third party. In the second step local data owners train with their local private data and obtain an updated model of weights and gradients corresponding to their unique private data. In the last step each local data owner sends their updated results back to the trusted third party which combines and averages the updates to form the new global model. This new global model can be shared back to the parties for future training updates and use during inference.

Traditional federated learning suggests an optimal solution to the privacy issue as no local data is shared between joint computation members. Past research shows that sharing updated weights and gradients can open leakage attacks which enable methods of obtaining individual private data (Rahimian et al., 2021). In addition, the requirement of a trust third party to maintain security during the federated learning process places a single point of penetration to the entire system.

A major research focus is taking the base federated learning algorithm and integrated methods of encryption and/or secure computation as a method to secure the transfer of model updates, also known as *secure aggregation*. Secure aggregation methods include use of differential privacy (Byrd and Polychroniadou, 2020), homomorphic encryption (Truex et al., 2019; Xu et al., 2019), and multi-party computation (Bonawitz et al., 2017). Each provides different methods of adding security to global

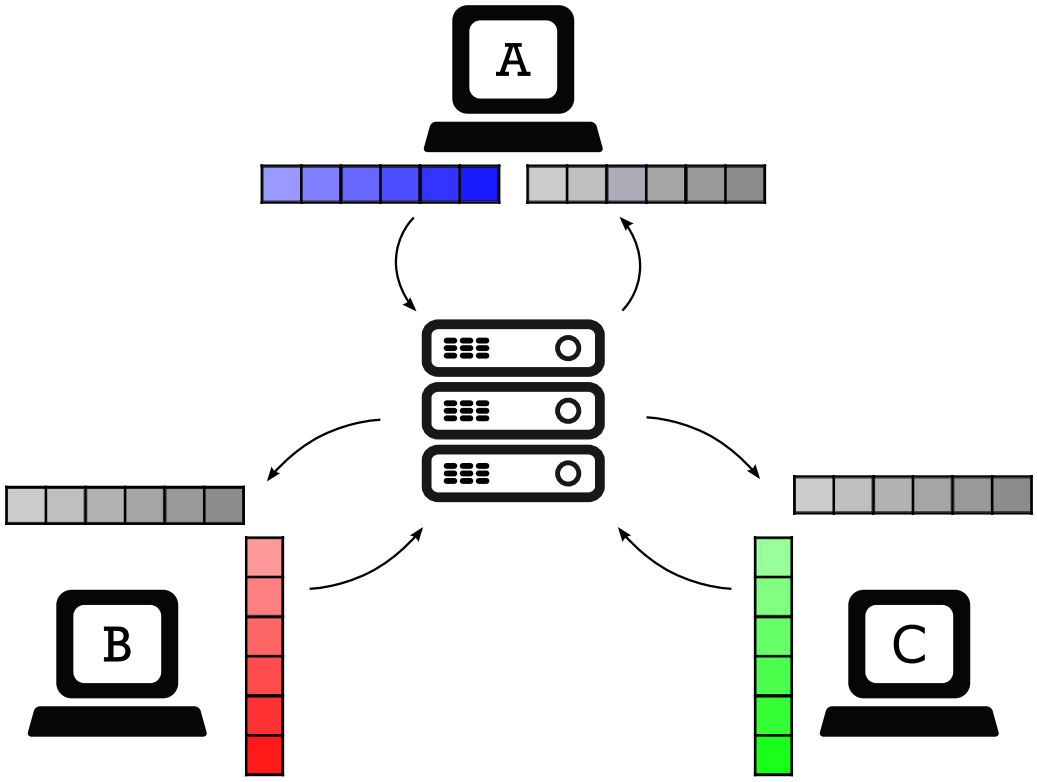


Figure 6·17: Model update method for classic Federated Learning requiring trusted third party. Models are initial obtained from the global server and each local system performs training updates based on local data. Upon completion, models are shared back to global third party which performs a federated averaging function to generate the new global model.

model updates at the cost of accuracy and/or performance.

Background and Related Work

Federated learning can take on forms such as Cross-device, Cross-silo, and many others (Kairouz et al., 2021). Each federated learning environment may introduce different problems which guide implementation of privacy and security in different directions.

Cross-device federated learning consists of many devices (e.g. mobile or IoT) each possessing unique private data which is used as input into the training of a centralized ML algorithm. Private data is never shared and devices create local updates to a

global model and share their updates with a central server for global averaging. On completion of the training, each device is updated with the new model from the central server before participating again in future training updates. This form of federated learning can utilize hundreds or more low power devices each performing local ML updates and sharing their updates with a server over a traditionally slower network connection.

Cross-silo federated learning uses a smaller number of contributing parties each possessing a larger amount of data typically consolidated from many sources (e.g., in healthcare and finance). Private data remains local to each member. However, larger model updates may be performed over many more data points. Global model updates may be assigned and computed by a trusted third party and eventually shared with all members, or jointly computed between all contributing members.

Privacy Solutions

Federated learning methods require the use of cryptographically secure aggregation methods to enable global updates without data leakage. Secure aggregation can be performed through a few different methods which utilize encryption and obfuscation to share ciphertext of locally generated model weights and gradients. The construction of the global machine learning model is then done by combining the ciphertext values with the original model through methods such as federated averaging or federated stochastic gradient descent. Upon completion plaintext values for each global weight is obtained through decryption or share combination methods. In the next subsections, we discuss the different methods of secure aggregation and the limitations involved with each method.

- Differential Privacy (DP) - Differential privacy is the method of obfuscating local model updates during secure aggregation methods through the use of small

statistically insignificant additions which do not cause a global change of the generated model. Traditionally Gaussian noise is added to the weights of each local model update prior to performing secure aggregation (Wei et al., 2020). The addition of noise to the local model updates may help to prevent back tracing of private data contributions. The best case scenario for differential privacy is with large number of clients in a cross-device federated learning environment as each inclusion of noise may only introduce small inaccuracies toward the global model.

- Fully Homomorphic Encryption (FHE) - Fully Homomorphic Encryption uses algorithmic techniques to perform data computation on encrypted ciphertext which upon completion can be decrypted by the data owner to obtain the final result. Based on federated learning requirements, encryption keys and ciphertext can be shared and generated by clients or the server performing the secure aggregation step. One option the aggregation server performs all HE functions including decryption of the final values generated the final global model updates locally and transfers the final results to each collaborating member. The second option, each private data holder encrypts their local model updates and shares these updates with an aggregation server. The server performs the secure aggregation and returns the ciphertext version of the global update model back to the private data holders, which decrypt the new values and update their local model with global parameters. This second method enables an untrusted third party aggregation server to be central point of contact between contributors which can utilize a high number of compute resources. FHE algorithms require a large amount of computation resources resulting in overall low performance during secure aggregation. Acceleration of FHE continues to be a common research topic utilizing distributed resources, GPU and FPGA acceler-

ation, and improving upon algorithmic techniques through data packing. FHE can be a viable solution to secure aggregation, however with low performance and demanding hardware requirements alternative methods may provide better options if applicable.

- Multi-Party Computation (MPC) - Multi-Party Computation typically is performed with a small number of contributing party members (e.g. 2 party, 3 party, or 4 party). MPC is a method of performing agreed upon computation between individuals with shared private data. Data owners first start by splitting data into shares and distributing them between party members each involved with the joint computation. Each share does not contain enough information to reconstruct the initial data, but allows for computation to be performed both locally and jointly between members to obtain a shared result. Federated learning with MPC enables all contributing members to jointly compute the global model update and only requires a single round of communication upon completion to obtain plaintext results of the new model weights.

Proposed Datacenter Model Design

The environment and available resources in which federated learning takes place can heavily dictate the best privacy solution for the optimal performance. The datacenter continues to grow and attract more users through fully managed and on-demand resources which enables short term large application computation. Increasing number of organizations have migrated most of their operations to the datacenter including hosting their data and utilizing the available compute nodes. This includes those with protected data as more options for private cloud instances are ever increasing allowing access to datacenter resources in a secure manner. We believe a model where organizations utilize datacenter resources to compute on private data as shown

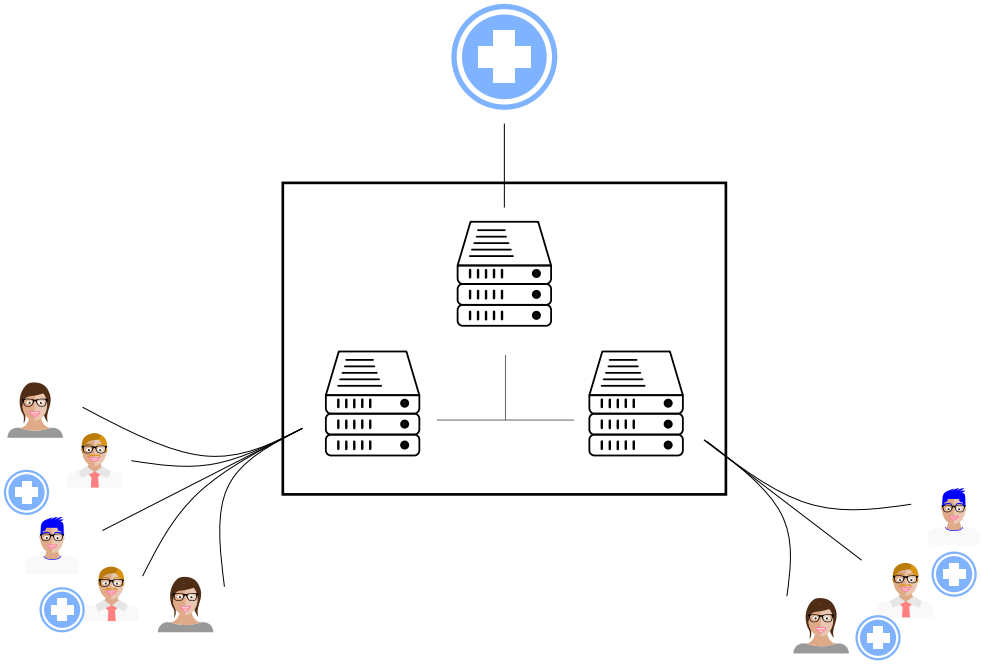


Figure 6·18: Shows a potential environment where users share their private data with an organization data silos that collects and stores this data for joint computation in the future.

in Figure 6·18. This model allows for individual data owners to coordinate with an institution such as a healthcare system to maintain secure possession of data. Organizations are free to utilize any method of obtaining private data from sources and maintain full access to this data for future processing. Multiple organizations utilizing the same datacenter resources may agree to secure joint computation over their private data in a federated learning scenario. Thus we believe it would not be unreasonable to suggest parties may agree to collocate in the same datacenter to obtain the best network performance when performing joint computation. With the above assumptions, we believe MPC will provide the best method to privacy aware joint computation in the datacenter setting as MPC can utilize the performance benefits for both computation and communication when operating on a federated learning application.

With the help of datacenter resources and secure private enclaves, we envision a design where there is no global model holder and instead each data holder organization jointly performs the secure aggregation steps with all other parties. Removing the single model aggregator eliminates a single point of attack in the system and helps to keep all parties semi-honest with each other for mutual benefit. We list out the steps required for a single epoch below:

1. Collect data - As a data holder, when new confidential data is available collect and store securely. This can be done with traditional encryption processes or through any secure data transfer methods. Using private enclaves and encrypted data storage, all confidential data will only be held in encrypted form and accessible by private key holders by the trusted organization.

2. Perform addition training with new data - Each data holder has access to the global model at all times and can perform additional rounds of training utilizing any number of datacenter resources. Differential privacy is used during the

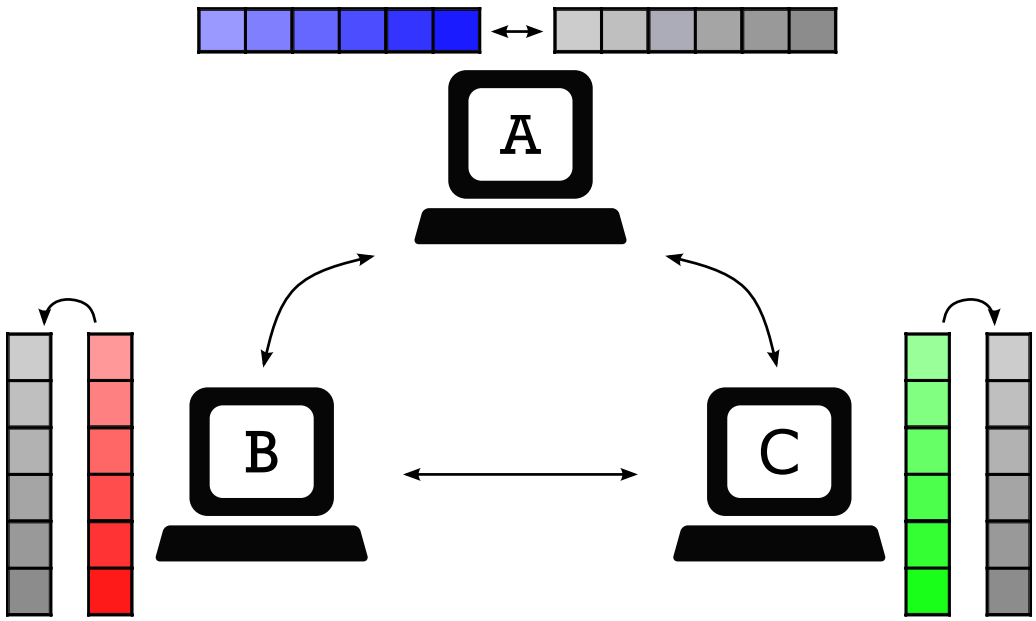


Figure 6·19: Model update method for proposed MPC version of Federated Learning. Each participating member locally owns the global version of the model. Each local system performs training updates based on local data. Upon completion, models are encrypted into shares and distributed among the group. Secure aggregation is jointly performed and collectively final models are generated upon completion and stored in plain-text.

training process to help create an additional level of security against back tracing after model updates are shared among other party members.

3. Indicate to party members of new global update - Once all local training is complete, party members can indicate to the group of semi-honest individuals that a new model is generated and can be shared. No data is shared at this time and there is no limit to the number of party members with an update to contribute.

4. Secure joint computation of model update - Here we perform the procedure of secure aggregation using MPC. The global model is stored in both MPC share form and plain-text form, which helps to make future joint model updates easier and local training updates quicker. Using the MPC share form of the global model, each party with local changes creates new shares of their local weights or gradients and distributes them to all the cooperating party members. Once completed, each party member will have a piece of information from all member updates and can then begin the process of secure aggregation. Parties jointly compute the updates to the MPC share global model and upon completion each have a piece of the global model update. Last the final computation step is performed to create a new plain-text version of the global model for future epochs and local inference.

5. Local inference against new global model - Each party member can perform local inference with the plain-text model and does not require any communication with other party members.

6.6 Summary

Multi-Party Computation aims to be a viable option for privacy preserving computation between privacy aware organizations. However, due to the complexity of current algorithms, performance is limited by both CPU and network fabric. In this chapter, we propose the use of network connected FPGAs in the datacenter to accelerate the performance of secret sharing MPC.

Our FPGA implementations for 3PC and 4PC algorithms Araki et al. (Araki et al., 2016b) and Fantastic Four (Dalskov et al., 2021) respectively use only 3% of FPGA resources and obtain more throughput compared to the original CPU designs. Using the COPA lookaside accelerator framework, we can run multiple accelerators in parallel and fully utilize the 100Gb/s network resource to perform over 6000 MPC operations per second.

We aim to incorporate our MPC hardware accelerators into a fully featured MPC Machine Learning environment. In our initial experiments running machine learning functions such as matrix multiply and ReLU activation, we find significant improvements due to both parallel MPC accelerators and high bandwidth connections. In addition, we propose a unique alternative method for Federated Learning which removes the requirement of a third party aggregator and allows all parties to have full control over the global training process. Combining our FPGA accelerated MPC with distributed Machine Learning demonstrates the full capabilities of our design for a real world application scenario.

Chapter 7

Conclusion and Future Work

In this chapter, we summarize the contributions of this thesis, discuss the future direction of research in the field of security of datacenters for classified use, and outline generally the directions for future work of FPGA SmartNICS in the datacenter, especially for security.

7.1 Summary of Major Contributions

Datacenter security continues to be an important research field as a growing number of organizations require additional security guarantees to use public cloud resources. In this dissertation, we propose various methods of improving the performance and security of the datacenter for tenants with confidential data requirements. In our proposed methods, we leverage FPGAs as they provide an ideal acceleration platform for various types of applications, especially for those with high communication requirements. Using FPGA SmartNICs, we improve the performance of datacenter operations for both tenant applications and cloud infrastructure.

With recent advances in datacenters, cloud providers leverage FPGAs SmartNICs to provide performance improvements for distributed applications. Such FPGA SmartNICs can act as co-processor accelerators and headless remote accelerators. We demonstrate that offloading common network functions to FPGA SmartNICs reduces the load of CPUs on each datacenter node and improves the overall bandwidth and network congestion. As an example, we implement a hardware packet fragmentation

offload engine which operates in the pipeline of the COPA TX network. Leveraging our proposed fragmentation offload engine, we can write the software without knowing the network MTU requirements. At the hardware level, each fragmentation packet is treated as an individual packet on the network. Subsequently, individual fragments have separate confirmation of success and retries, which improves the reliability and reduces the bandwidth requirements. In addition to network function offload, the FPGA provides additional resources for tenant application offload in the form of inline and lookaside acceleration. Inline packets provide custom acceleration options which can be chained together and operate on the payload of data during transmit and receive of each packet. Lookaside acceleration on an FPGA SmartNIC enables a similar co-process system with the added capabilities of using the high bandwidth network resources without CPU intervention.

In addition, FPGAs attached to cloud resources enable security improvements through separation of tenant applications and cloud management features. This hardware separation allows for advanced control of node access and network connections. We demonstrate the ability to follow strict government communication protocols known as CSfC with the use of a custom hardware encryption environment enabled through FPGA SmartNICs. With hardware attestation features such as FSS and root of trust zones, tenants have access to security features without needing additional trust from cloud providers. Reconfigurable regions on the FPGA enable tenants to deploy custom hardware encryption and advanced cryptographic algorithms.

Finally, we propose the network attached FPGAs in the datacenter as an ideal environment for high performance secret sharing Multi-Party Computation. High bandwidth and low latency networks available on FPGA SmartNICs allow secret sharing MPC to achieve high performance compared to current implementations. Our results indicate that current MPC implementations are limited by CPU and NIC

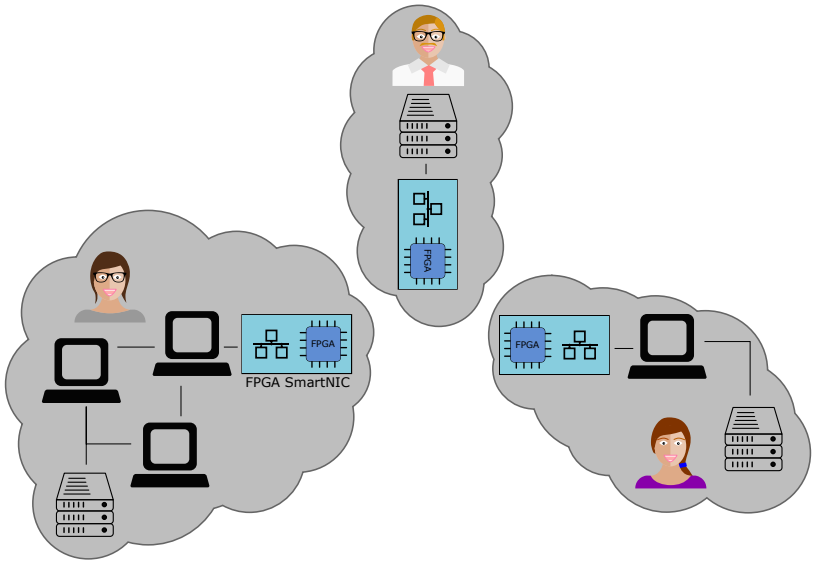
performance whereas network attached FPGAs provided an optimal solution to both computation and communication bottlenecks. Our implementation can effectively saturate traditional 10Gb/s networks while only using a fraction of the total available FPGA hardware resources. Implementations running as a lookaside accelerator on COPA can achieve upto 17.5Gb/s of MPC operations and we only require 6 parallel accelerators to fully utilize the 100Gb/s COPA network. A single MPC lookaside accelerator uses less than 6% of the total FPGA resources.

7.2 Future Research Directions

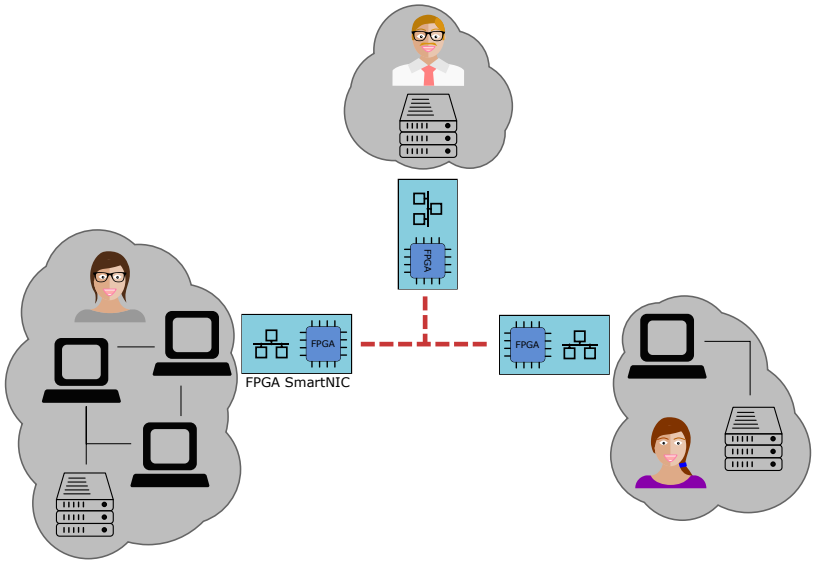
In this section, we outline the potential directions for future work. Our vision is to enable secure public clouds for confidential data processing or what we call MPC as a Service (MPCaaS).

7.2.1 Secure MPC Between Private Cloud Enclaves

As a future direction, we propose a framework that uses headless FPGA SmartNICs to enable a secure communication channel between private cloud enclaves. Such secure communication channels allow us to perform confidential data processing. In this framework, FPGAs in the datacenter create a temporary private network between parties performing secure joint computation as shown in Figure 7·1. Each temporary network can be created and destroyed on demand. To maintain data privacy, these temporary networks never interact with tenant private cloud enclaves and their associated nodes. To eliminate the possibility of learning the original plain-text data outside of the secure enclave, all the data stored on each FPGA will be in the form of secret shares. While performing the computation, parties are given a choice to communicate over the network with non-encrypted shares of MPC data for the best performance, or use hardware encryption algorithms running as inline accelerators to increase data security on the temporary network. In the latter scenario, parties enable



(a) Three Private Secure Enclaves



(b) Temporary MPC Network between FPGA SmartNICs

Figure 7·1: Tenants have their own secure private enclaves keeping classified data secure from others in tenants in the public cloud. While performing MPC a temporary private network is created connecting each party's FPGA SmartNIC together. Once complete, the nodes are returned to each private enclave with the final data.

point-to-point security by sharing unique key pairs with each member. This approach is similar to the system described in subsection 4.3.3 and limits the possibility of man-in-the-middle attacks. To achieve optimal performance, MPC is implemented using lookaside accelerators as demonstrated in section 6.4. Finally upon completion, these temporary networks would be terminated to avoid additional opportunities of data leakage.

To achieve our proposed solution, we can employ cloud infrastructure functions on both FPGA SmartNICs and top-of-rack switches to create temporary networks similar to HIL (Hennessey et al., 2016). Additionally, the inclusion of the remote invocation feature from COPA provides an opportunity to automate the computation between each party member. The combination of these features would enable a fully autonomous MPC operation between agreeing parties.

7.2.2 Extending MPC Hardware Acceleration to PaaS

So far, we have focused our efforts on MPC improvements through dedicated hardware accelerators; however, we have neglected a very important aspect of high performance computing. An important aspect of our proposed future direction should focus on the interaction between software libraries and our hardware accelerators. The goal is to enable users, such as data scientists, to leverage the benefits of secure joint computation without the need to have a full understanding of the underlying hardware system. To this end, we envision providing systems similar to Platform-as-a-Service (PaaS) offerings for machine learning in the datacenter. Software development tools such as PyTorch for machine learning offer development platforms for specialized hardware systems including transparent integration with GPUs. There are a myriad of software libraries in development to support MPC algorithms and frameworks, with many focused on distributed machine learning as their target application (Zhao et al., 2019; Ohrimenko et al., 2016; Du and Atallah, 2001; Knott et al., 2020).

One avenue to achieve an integrated software and hardware experience is to include accelerator additions directly into the currently offered MPC software packages. For this to be successful, a standardized hardware communication layer is required to provide a method of interaction with all types of applications for hardware accelerators. OFI libfabric provides an open source software layer to communicate and interact with a number of different network communication accelerators. Combining this open source framework with open source MPC software libraries would allow for continued improvement of the communication layers and neither side would be stifled by a closed source ecosystem.

Eventually, the goal is to enable a fully complete Software-as-a-Service (SaaS), which enables organizations with little to no knowledge of software development to utilize cloud offerings. Subsequently, such organizations can securely manage and operate on confidential data without the fear of data leakage during joint computation with others.

7.3 Final Remarks

In summary, we propose multiple methods of providing classified data holders the opportunity to utilize public cloud resources for confidential data processing. We demonstrate how FPGA SmartNICs provide great value to datacenter growth, application performance, and mechanisms for public cloud security. The inclusion of FPGAs in the datacenter is not limited to cloud provider operations, but can be used by tenants to secure the network and improve the performance of secure joint computation. We strongly believe that FPGA SmartNICs in the public cloud offer substantial possibilities for security and performance and can be used by all organizations to protect confidential data and eliminate future data leaks.

References

Abidin, A., Aly, A., Cleemput, S., and Mustafa, M. A. (2016). An MPC-Based Privacy-Preserving Protocol for a Local Electricity Trading Market. In *15th International Conference on Cryptology and Network Security (CANS)*, volume 10052 of *Lecture Notes in Computer Science*, pages 615–625.

Agrawal, D., Archambeault, B., Rao, J. R., and Rohatgi, P. (2003). The em side—channel(s). In Kaliski, B. S., Koç, ç. K., and Paar, C., editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 29–45, Berlin, Heidelberg. Springer Berlin Heidelberg.

Amazon (2016). aws_fpga. `https://github.com/aws/aws-fpga.git`.

Amazon (2022a). Amazon EC2 F1 instances. `https://aws.amazon.com/ec2/instance-types/f1/`.

Amazon (2022b). Amazon EC2 Instance Types. `https://aws.amazon.com/ec2/instance-types/`.

Amazon (2022c). AWS GovCloud (US). `https://aws.amazon.com/govcloud-us/`.

Amazon (2022d). AWS Nitro System. `https://aws.amazon.com/ec2/nitro/`.

Araki, T., Barak, A., Furukawa, J., Lichter, T., Lindell, Y., Nof, A., Ohara, K., Watzman, A., and Weinstein, O. (2017). Optimized Honest-Majority MPC for Malicious Adversaries - Breaking the 1 Billion-Gate per Second Barrier. *Proceedings - IEEE Symposium on Security and Privacy*, pages 843–862.

Araki, T., Barak, A., Furukawa, J., Lindell, Y., Nof, A., and Ohara, K. (2016a). DEMO: High-Throughput Secure Three-Party Computation of Kerberos Ticket Generation. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1841–1843, New York, NY, USA. Association for Computing Machinery.

Araki, T., Furukawa, J., Lindell, Y., Nof, A., and Ohara, K. (2016b). High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 805–817, New York, NY, USA. Association for Computing Machinery.

Archer, D. W., Bogdanov, D., Lindell, Y., Kamm, L., Nielsen, K., Pagter, J. I., Smart, N. P., and Wright, R. N. (2018). From Keys to Databases - Real-World Applications of Secure Multi-Party Computation. *The Computer Journal*, 61(12):1749–1771.

AWS Public Sector Blog Team (2017). Announcing the new AWS Secret Region. `https://aws.amazon.com/blogs/publicsector/announcing-the-new-aws-secret-region/`.

Barr, J. (2017). Amazon EC2 bare metal instances with direct access to hardware. `https://aws.amazon.com/blogs/aws/new-amazon-ec2-bare-metal-instances-with-direct-access-to-hardware/`.

Beaver, D., Micali, S., and Rogaway, P. (1990). The round complexity of secure protocols. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 503–513, New York, NY, USA. Association for Computing Machinery.

Bogdanov, D., Kamm, L., Kubo, B., Rebane, R., Sokk, V., and Talviste, R. (2016). Students and taxes: a privacy-preserving study using secure computation. *Proceedings on Privacy Enhancing Technology (PoPETs)*, 2016(3):117–135.

Bogetoft, P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J. D., Nielsen, J. B., Nielsen, K., Pagter, J., Schwartzbach, M., and Toft, T. (2009). Secure multiparty computation goes live. In Dingledine, R. and Golle, P., editors, *Financial Cryptography and Data Security*, pages 325–343, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA. Association for Computing Machinery.

Byali, M., Chaudhari, H., Patra, A., and Suresh, A. (2020). Flash: fast and robust framework for privacy-preserving machine learning. *Proceedings on Privacy Enhancing Technologies*, 2020(2):459–480.

Byrd, D. and Polychroniadou, A. (2020). Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA. Association for Computing Machinery.

Castillo, J. (2004). systemc_rng. `https://opencores.org/projects/systemc_rng`.

Caulfield, A., Chung, E., Putnam, A., Angepat, H., Fowers, J., Haselman, M., Heil, S., Humphrey, M., Kaur, P., Kim, J.-Y., Lo, D., Massengill, T., Ovtcharov, K., Papamichael, M., Woods, L., Lanka, S., Chiou, D., and Burger, D. (2016). A cloud-scale acceleration architecture. In *49th IEEE/ACM Int. Symp. Microarchitecture*, pages 1–13.

Chaudhari, H., Rachuri, R., and Suresh, A. (2020). Trident: Efficient 4pc framework for privacy preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020*.

Cheon, J. H., Kim, A., Kim, M., and Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In Takagi, T. and Peyrin, T., editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham. Springer International Publishing.

Chiu, M. and Herbordt, M. (2009). Efficient filtering for molecular dynamics simulations. In *2009 International Conference on Field Programmable Logic and Applications*. doi: 10.1109/ FPL15426.2009.

Chiu, M. and Herbordt, M. (2010). Molecular dynamics simulations on high performance reconfigurable computing systems. *ACM Transactions on Reconfigurable Technology and Systems*, 3(4):1–37. doi: 10.1145/1862648.1862653.

Chiu, M., Khan, M., and Herbordt, M. (2011). Efficient calculation of pairwise nonbonded forces. In *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*. doi: 10.1109/ FCCM.2011.34.

Chung, E., Fowers, J., Ovtcharov, K., Papamichael, M., Caulfield, A., Massengill, T., Liu, M., Lo, D., Alkalay, S., Haselman, M., et al. (2018). Serving DNNs in real time at datacenter scale with Project Brainwave. *IEEE Micro*, 38(2):8–20.

Cisco (2022). Cisco Nexus SmartNIC. `https://www.cisco.com/c/en/us/products/interfaces-modules/nexus-smartnic/index.html`. Last accessed on 2022-07-18.

Colbert, K. (2020). Announcing Project Monterey – redefining hybrid cloud architecture. `https://blogs.vmware.com/vsphere/2020/09/announcing-project-monterey-redefining-hybrid-cloud-architecture.html`.

Conti, A., VanCourt, T., and Herbordt, M. (2004). Processing Repetitive Structures with Mismatches at Streaming Rate. In *Field Programmable Logic and Application. FPL 2004. Lecture Notes in Computer Science, vol 3203. Springer, Berlin, Heidelberg*. doi: 10.1007/978-3-540-30117-2_131.

Dalskov, A., Escudero, D., and Keller, M. (2021). Fantastic four: Honest-Majority Four-Party secure computation with malicious security. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2183–2200. USENIX Association.

Damgård, I., Damgård, K., Nielsen, K., Nordholt, P. S., and Toft, T. (2017). Confidential benchmarking based on multiparty computation. In Grossklags, J. and Preneel, B., editors, *Financial Cryptography and Data Security*, pages 169–187, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Demmler, D., Schneider, T., and Zohner, M. (2015). ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *2015 Network and Distributed System Society (NDSS) Symposium.* The Internet Society.

Du, P., Weber, R., Luszczek, P., Tomov, S., Peterson, G., and Dongarra, J. (2012). From CUDA to OpenCL: Towards a performance-portable solution for multi - platform GPU programming. *Parallel Computing*, 38(8):391–407. Application Accelerators in HPC.

Du, W. and Atallah, M. J. (2001). Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 Workshop on New Security Paradigms*, NSPW '01, page 13–22, New York, NY, USA. Association for Computing Machinery.

Eran, H., Zeno, L., Tork, M., Malka, G., and Silberstein, M. (2019). NICA: An infrastructure for inline acceleration of network applications. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 345–362, Renton, WA. USENIX Association.

Evans, D., Kolesnikov, V., and Rosulek, M. (2018). *A Pragmatic Introduction to Secure Multi-Party Computation.* NOW Publishers.

Fang, X., Ioannidis, S., and Leeser, M. (2017). Secure function evaluation using an FPGA overlay architecture. *FPGA 2017 - Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 257–266.

Fang, X., Ioannidis, S., and Leeser, M. (2019). SIFO: secure computational infrastructure using FPGA overlays. *International Journal of Reconfigurable Computing*, 2019:1439763.

Feigenbaum, J., Pinkas, B., Ryger, R., and Saint-Jean, F. (2004). Secure computation of surveys. In *EU Workshop on Secure Multiparty Protocols*, pages 2–14.

Firestone, D., Putnam, A., Mundkur, S., Chiou, D., Dabagh, A., Andrewartha, M., Angepat, H., Bhanu, V., Caulfield, A., Chung, E., Chandrappa, H. K., Chaturmohta, S., Humphrey, M., Lavier, J., Lam, N., Liu, F., Ovtcharov, K., Padhye, J., Popuri, G., Raindel, S., Sapre, T., Shaw, M., Silva, G., Sivakumar, M., Srivastava, N., Verma, A., Zuhair, Q., Bansal, D., Burger, D., Vaid, K., Maltz, D. A., and Greenberg, A. (2018). Azure accelerated networking: SmartNICs in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 51–66, Renton, WA. USENIX Association.

Fornero, M., Maunero, N., Prinetto, P., and Varriale, A. (2020). SEkey: a distributed hardware-based key management system. In *2020 IEEE East-West Design & Test Symposium (EWDTS)*, pages 1–7.

Fraunholz, D., Krohmer, D., Duque Anton, S., and Schotten, H. D. (2018). Catch Me If You Can: dynamic concealment of network entities. In *Proceedings of the 5th ACM Workshop on Moving Target Defense*, MTD '18, page 31–39, New York, NY, USA. Association for Computing Machinery.

Frederiksen, T. K., Jakobsen, T. P., and Nielsen, J. B. (2014). Faster maliciously secure two-party computation using the GPU. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8642(grant 61061130540):358–379.

Fungible (2022). The Fungible data processing unit. `https://www.fungible.com/product/dpu-platform/`.

Furukawa, J. and Lindell, Y. (2019). Two-thirds honest-majority MPC for malicious adversaries at almost the cost of semi-honest. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1557–1571.

Furukawa, J., Lindell, Y., Nof, A., and Weinstein, O. (2017). High-throughput secure three-party computation for malicious adversaries and an honest majority. In Coron, J.-S. and Nielsen, J. B., editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 225–255, Cham. Springer International Publishing.

Gainaru, A., Graham, R. L., Polyakov, A., and Shainer, G. (2016). Using InfiniBand hardware gather-scatter capabilities to optimize MPI All-to-All. In *Proceedings of the 23rd European MPI Users' Group Meeting*, EuroMPI 2016, page 167–179, New York, NY, USA. Association for Computing Machinery.

Geng, T. (2021). *FPGA-Based High-Performance Neural Network Acceleration*. PhD thesis, Department of Electrical and Computer Engineering, Boston University. ProQuest Number: 28259172.

Geng, T., Li, A., Shi, R., Wu, C., Wang, T., Li, Y., Haghi, P., Tumeo, A., Che, S., Reinhardt, S., and Herbordt, M. (2020a). AWB-GCN: A Graph Convolutional Network Accelerator with Runtime Workload Rebalancing. In *53rd IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

Geng, T., Wang, T., Sanaullah, A., Yang, C., Xuy, R., Patel, R., and Herbordt, M. (2018a). A framework for acceleration of CNN training on deeply-pipelined FPGA clusters with work and weight load balancing. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL 2018): 394–402.* doi: 10.1109/ FPL.2018. 00074.

Geng, T., Wang, T., Sanaullah, A., Yang, C., Xuy, R., Patel, R., and Herbordt, M. (2018b). FPDeep: Acceleration and Load Balancing of CNN Training on FPGA Clusters. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, page 81–84. doi: 10.1109/ FCCM.2018. 00021.

Geng, T., Wang, T., Wu, C., Li, Y., Yang, C., Wu, W., Li, A., and Herbordt, M. (2021a). O3BNN-R: An Out-Of-Order Architecture for High-Performance and Regularized BNN Inference. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):199–213. doi: 10.1109/TPDS.2020.3013637.

Geng, T., Wang, T., Wu, C., Yang, C., Li, A., Song, S., and Herbordt, M. (2019a). LP-BNN: Ultra-low-Latency BNN Inference with Layer Parallelism. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, volume 2160, pages 9–16. doi: 10.1109/ASAP.2019.00-43.

Geng, T., Wang, T., Wu, C., Yang, C., Wu, W., Li, A., and Herbordt, M. (2019b). O3BNN: An Out-Of-Order Architecture for High-Performance Binarized Neural Network Inference with Fine-Grained Pruning. In *ACM International Conference on Supercomputing*, volume 2160, pages 461–472. doi: 10.1145/ 3330345. 3330386.

Geng, T., Wu, C., Tan, C., Fang, B., Li, A., and Herbordt, M. (2020b). CQNN: a CGRA-based QNN Framework. In *IEEE High Performance Extreme Computing Conference*. doi: 10.1109/HPEC43674.2020.9286194.

Geng, T., Wu, C., Tan, C., Xie, C., Guo, A., Haghi, P., He, S., Li, J., Herbordt, M., and Li, A. (2021b). A Survey: Handling Irregularities in Neural Network Acceleration with FPGAs. In *IEEE High Performance Extreme Computing Conference*. doi:10.1109/HPEC49654.2021.9622877.

Geng, T., Wu, C., Zhang, Y., Tan, C., Xie, C., You, H., Herbordt, M., Lin, Y., and Li, A. (2021c). I-GCN: A Graph Convolutional Network Accelerator with Runtime Locality Enhancement Through Islandization. In *54th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. doi:10.1145/3466752.3480113.

George, A., Herbordt, M., Lam, H., Lawande, A., Sheng, J., and Yang, C. (2016). Novo-G#: A Community Resource for Exploring Large-Scale Reconfigurable Computing Through Direct and Programmable Interconnects. In *2016 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA*, pages 1–7. doi: 10.1109/HPEC.2016.7761639.

Ghafir, I., Prenosil, V., Svoboda, J., and Hammoudeh, M. (2016). A survey on network security monitoring systems. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 77–82.

Giannopoulos, T. and Mouris, D. (2018). *Privacy Preserving Medical Data Analytics using Secure Multi Party Computation. An End-To-End Use Case.* PhD thesis, National and Kapodistrian University of Athens.

Grant, S., Yelam, A., Bland, M., and Snoeren, A. C. (2020). SmartNIC performance isolation with FairNIC: Programmable networking for the cloud. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 681–693, New York, NY, USA. Association for Computing Machinery.

Grun, P., Hefty, S., Sur, S., Goodell, D., Russell, R. D., Pritchard, H., and Squyres, J. M. (2015). A brief introduction to the OpenFabrics interfaces - a new network API for maximizing high performance application efficiency. In *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, pages 34–39.

Gu, J., Wang, J., Yu, Z., and Shen, K. (2018). Walls have ears: Traffic-based side-channel attack in video streaming. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1538–1546.

Gu, Y. and Herbordt, M. (2007). FPGA-based multigrid computations for molecular dynamics simulations. In *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 117–126. doi: 10.1109/ FCCM.2007.42.

Gu, Y., VanCourt, T., and Herbordt, M. (2006). Integrating FPGA acceleration into the ProtoMol molecular dynamics code: Preliminary report. In *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 315–316. doi: 10.1109/ FCCM.2006.52.

Gu, Y., VanCourt, T., and Herbordt, M. (2008). Explicit design of FPGA-based coprocessors for short-range force computation in molecular dynamics simulations. *Parallel Computing*, 34(4-5):261–271. doi: 10.1016/j.parco.2008.01.007.

Haghi, P., Geng, T., Guo, A., Wang, T., and Herbordt, M. (2020a). Reconfigurable Compute-in-the-Network FPGA Assistant for High-Level Collective Support with

Distributed Matrix Multiply Case Study. In *IEEE Conference on Field Programmable Technology*. doi: 10.1109/ICFPT51103.2020.00030.

Haghi, P., Guo, A., Geng, T., Skjellum, A., and Herbordt, M. (2021). Workload Imbalance in HPC Applications: Effect on Performance of In-Network Processing. In *IEEE High Performance Extreme Computing Conference*. doi: 10.1109/HPEC49654.2021.9622847.

Haghi, P., Guo, A., Xiong, Q., Patel, R., Yang, C., Geng, T., Broaddus, J., Marshall, R., Skjellum, A., and Herbordt, M. (2020b). FPGAs in the Network and Novel Communicator Support Accelerate MPI Collectives. In *IEEE High Performance Extreme Computing Conference*. doi: 10.1109/HPEC43674.2020.9286200.

Haghi, P., Guo, A., Xiong, Q., Yang, C., Geng, T., Broaddus, J., Marshall, R., Schafer, D., Skjellum, A., and Herbordt, M. (2022). Reconfigurable switches for high performance and flexible mpi collectives. *Concurrency and Computation: Practice and Experience*, 34(2). doi: 10.1002/cpe.6769.

Hennessey, J., Tikale, S., Turk, A., Kaynar, E. U., Hill, C., Desnoyers, P., and Krieger, O. (2016). Hil: Designing an exokernel for the data center. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, SoCC '16, page 155–168, New York, NY, USA. Association for Computing Machinery.

Herbordt, M. (2013). Architecture/algorithm codesign of molecular dynamics processors. In *2013 Asilomar Conference on Signals, Systems, and Computers*, pages 1442–1446. doi: 10.1109/ ACSSC.2013.6810534.

Herbordt, M., Gu, Y., VanCourt, T., Model, J., Sukhwani, B., and Chiu, M. (2008). Computing models for FPGA-based accelerators with case studies in molecular modeling. *Computing in Science and Engineering*, 10(6):35–45. doi: 10.1109/ MCSE.2008.143.

Herbordt, M. and VanCourt, T. (2005). System and method for programmable logic acceleration of data processing applications and compiler therefore. United States Patent Application Publication. Pub. no.: US2007/0277161 A1.

Herbordt, M., VanCourt, T., Gu, Y., Sukhwani, B., Conti, A., Model, J., and DiSabello, D. (2007). Achieving high performance with FPGA-based computing. *IEEE Computer*, 40(3):42–49.

Hsing, H. (2012). tiny_aes. `https://opencores.org/projects/tiny_aes`.

Huang, K., Gungor, M., Fang, X., Ioannidis, S., and Leeser, M. (2019). Garbled circuits in the cloud using FPGA enabled nodes. *2019 IEEE High Performance Extreme Computing Conference, HPEC 2019*, pages 1–6.

Huang, Y., Evans, D., and Katz, J. (2012). Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? In *In 19th Network and Distributed Security Symposium (NDSS 2012).* The Internet Society.

Humphries, B., Zhang, H., Sheng, J., Landaverde, R., and Herbordt, M. (2014). 3D FFT on a Single FPGA. In *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines.* doi: 10.1109/ FCCM.2014.28.

Hund, R., Willems, C., and Holz, T. (2013). Practical timing side channel attacks against kernel space aslr. In *2013 IEEE Symposium on Security and Privacy*, pages 191–205.

Hussain, S. U. and Koushanfar, F. (2019). FASE: FPGA acceleration of secure function evaluation. *Proceedings - 27th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2019*, pages 280–288.

Hussain, S. U., Rouhani, B. D., Ghasemzadeh, M., and Koushanfar, F. (2018). MAXelerator: FPGA Accelerator for Privacy Preserving Multiply-Accumulate (MAC) on Cloud Servers. *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6.

Intel (2021). Intel unveils infrastructure processing unit. `https://www.intel.com/content/www/us/en/newsroom/news/infrastructure-processing-unit-data-center.html`.

Intel (2022a). Intel fpga smartnic n6000-pl platform. `https://www.intel.com/content/www/us/en/products/details/fpga/platforms/smartnic/n6000-pl-platform.html`. Last accessed on 2022-07-18.

Intel (2022b). Intel infrastructure processing unit (intel ipu) and smartnics. `https://www.intel.com/content/www/us/en/products/network-io/smartnic.html`. Last accessed on 2022-07-18.

Ion, M., Kreuter, B., Nergiz, A. E., Patel, S., Raykova, M., Saxena, S., Seth, K., Shanahan, D., and Yung, M. (2019). On Deploying Secure Computing Commercially: Private Intersection-Sum Protocols and their Business Applications. *IACR Cryptology ePrint Archive*, 2019:723.

Järvinen, K., Kolesnikov, V., Sadeghi, A. R., and Schneider, T. (2010a). Embedded SFE: Offloading server and network using hardware tokens. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6052 LNCS:207–221.

Järvinen, K., Kolesnikov, V., Sadeghi, A. R., and Schneider, T. (2010b). Garbled circuits for leakage-resilience: Hardware implementation and evaluation of one-time

programs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6225 LNCS:383–397.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210.

Keller, M. (2020). MP-SPDZ: A versatile framework for multi-party computation. Cryptology ePrint Archive, Report 2020/521. `https://eprint.iacr.org/2020/521`.

Kennedy, P. (2021). DPU vs SmartNIC and the STH NIC Continuum Framework. `https://www.servethehome.com/dpu-vs-smartnic-sth-nic-continuum-framework-for-discussing-nic-types`.

Khan, M., Chiu, M., and Herbordt, M. (2013). FPGA-Accelerated Molecular Dynamics. In Benkrid, K. and Vanderbauwhede, W., editors, *High Performance Computing Using FPGAs*, pages 105–135. Springer Verlag. doi: 10.1007/978-1-4614-1791-0_4.

Khan, M. and Herbordt, M. (2012). Communication requirements for FPGA-centric molecular dynamics. In *Symposium on Application Accelerators for High Performance Computing*. https:// www.bu.edu/ caadlab/saahpc12.pdf.

Kissner, L. and Song, D. (2005). Privacy-preserving set operations. In Shoup, V., editor, *Advances in Cryptology – CRYPTO 2005*, pages 241–257, Berlin, Heidelberg. Springer Berlin Heidelberg.

Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., and van der Maaten, L. (2020). CrypTen: Secure Multi-Party Computation Meets Machine Learning. In *Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning*.

Kolesnikov, V. and Schneider, T. (2008). Improved garbled circuit: Free XOR gates and applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5126 LNCS(PART 2):486–498.

Krishnan, V., Serres, O., and Blocksome, M. (2020). COnfigurable Network Protocol Accelerator (COPA): An Integrated Networking/Accelerator Hardware/Software Framework. In *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 17–24, Los Alamitos, CA, USA. IEEE Computer Society.

Krishnan, V., Serres, O., and Blocksome, M. (2021). Configurable Network Protocol Accelerator (COPA). *IEEE Micro*, 41(1):8–14.

Landaverde, R. and Herbordt, M. (2014). GPU Optimizations for a Production Molecular Docking Code. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*. doi: 10.1109/HPEC.2014.7040981.

Le, Y., Chang, H., Mukherjee, S., Wang, L., Akella, A., Swift, M. M., and Lakshman, T. V. (2017). UNO: Uniflying Host and Smart NIC Offload for Flexible Packet Processing. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, page 506–519, New York, NY, USA. Association for Computing Machinery.

Leeser, M., Gungor, M., Huang, K., and Ioannidis, S. (2019). Accelerating large garbled circuits on an FPGA-enabled cloud. *Proceedings of H2RC 2019: 5th International Workshop on Heterogeneous High-Performance Reconfigurable Computing - Held in conjunction with SC 2019: The International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 19–25.

Lenstra, A. K., Hughes, J. P., Augier, M., Bos, J. W., Kleinjung, T., and Wachter, C. (2012). Ron was wrong, whit is right. Cryptology ePrint Archive, Paper 2012/064. `https://eprint.iacr.org/2012/064`.

Li, B., Tan, K., Luo, L. L., Peng, Y., Luo, R., Xu, N., Xiong, Y., Cheng, P., and Chen, E. (2016). ClickNP: Highly flexible and high performance network processing with reconfigurable hardware. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 1–14, New York, NY, USA. Association for Computing Machinery.

Lin, L., Kasper, M., Güneysu, T., Paar, C., and Burleson, W. (2009). Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In Clavier, C. and Gaj, K., editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 382–395, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lindemann, J. and Fischer, M. (2018). A memory-deduplication side-channel attack to detect applications in co-resident virtual machines. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 183–192, New York, NY, USA. Association for Computing Machinery.

Ling, Z., Luo, J., Zhang, Y., Yang, M., Fu, X., and Yu, W. (2012). A novel network delay based side-channel attack: Modeling and defense. In *2012 Proceedings IEEE INFOCOM*, pages 2390–2398.

Liu, F., Yarom, Y., Ge, Q., Heiser, G., and Lee, R. B. (2015). Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy*, pages 605–622.

Mahram, A. and Herbordt, M. (2010). Fast and Accurate NCBI BLASTP: Acceleration with Multiphase FPGA-Based Prefiltering. In *24th ACM International Conference on Supercomputing*, pages 73–82. doi: 10.1145/1810085.1810099.

Mahram, A. and Herbordt, M. (2012). FMSA: FPGA-Accelerated ClustalW-Based Multiple Sequence Alignment through Pipelined Prefiltering. In *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, pages 177–183. doi: 10.1109/FCCM.2012.38.

Mahram, A. and Herbordt, M. (2015). NCBI BLASTP on High Performance Reconfigurable Computing Systems. *ACM Transactions on Reconfigurable Technology and Systems*, 15(4):6.1–6.20. doi: 10.1145/2629691.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*.

Miano, S., Doriguzzi-Corin, R., Risso, F., Siracusa, D., and Sommese, R. (2019). Introducing SmartNICs in server-based data plane processing: The DDoS mitigation use case. *IEEE Access*, 7:107161–107170.

Microsoft (2022). Azure for US Government. `https://azure.microsoft.com/en-us/global-infrastructure/government/`.

Model, J. and Herbordt, M. (2007). Discrete event simulation of molecular dynamics with configurable logic. In *2007 International Conference on Field Programmable Logic and Applications*, pages 151–158. doi: 10.1109/FPL.2007.4380640.

Mohassel, P. and Rindal, P. (2018). *ABY*^{3}: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 35–52, New York, NY, USA. Association for Computing Machinery.

Mohassel, P. and Zhang, Y. (2017). SecureML: a system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38.

Mosayyebzadeh, A., Mohan, A., Tikale, S., Abdi, M., Schear, N., Hudson, T., Munson, C., Rudolph, L., Cooperman, G., Desnoyers, P., and Krieger, O. (2019). Supporting security sensitive tenants in a Bare-Metal cloud. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 587–602, Renton, WA. USENIX Association.

Naor, M., Pinkas, B., and Sumner, R. (1999). Privacy preserving auctions and mechanism design. *ACM International Conference Proceeding Series*, pages 129–139.

National Security Agency (2022). Commercial solutions for classified program (csfc). https://www.nsa.gov/csfc/.

Netronome (2022). About Agilio SmartNICs. https://www.netronome.com/products/smartnic/overview/. Last accessed on 2022-07-18.

NIST (2022). Computer security resource center. https://csrc.nist.gov/glossary/term/High_Assurance_Internet_Protocol_Encryptor.

NVIDIA (2022a). ConnectX SmartNICs. https://www.nvidia.com/en-us/networking/ethernet-adapters/. Last accessed on 2022-07-18.

NVIDIA (2022b). NVIDIA BlueField Data Processing Units. https://www.nvidia.com/en-us/networking/products/data-processing-unit/. Last accessed on 2022-07-18.

NVIDIA (2022c). NVIDIA Mellanox Innova-2 Flex Open Programmable SmartNIC. https://www.nvidia.com/en-us/networking/ethernet/innova-2-flex/. Last accessed on 2022-06-5.

Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., and Costa, M. (2016). Oblivious multi-party machine learning on trusted processors. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 619–636, Austin, TX. USENIX Association.

Patel, R., Haghi, P., Jain, S., Kot, A., Krishnan, V., Varia, M., and Herbordt, M. (2022a). COPA Use Case: Distributed Secure Joint Computation. In *IEEE Symposium on Field Programmable Custom Computing Machines*. doi: 10.1109 / FCCM53951. 2022. 9786156.

Patel, R., Haghi, P., Jain, S., Kot, A., Krishnan, V., Varia, M., and Herbordt, M. (2022b). Distributed hardware accelerated secure joint computation on the COPA framework. *arXiv preprint arXiv:2204.04816*.

Patel, R., Wolfe, P.-F., Munafo, R., Varia, M., and Herbordt, M. (2020). Arithmetic and boolean secret sharing MPC on FPGAs in the data center. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. doi: 10.1109 / HPEC43674. 2020. 9286159.

Rahimian, S., Orekondy, T., and Fritz, M. (2021). Differential Privacy Defenses and Sampling Attacks for Membership Inference. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, AISec '21, page 193–202, New

York, NY, USA. Association for Computing Machinery. doi: 10.1145 / 3474369. 3486876.

Reinsel, D., Gantz, J., and Rydning, J. (2020). The digitization of the world from edge to core. `https://www.platinasystems.com/report-the-digitization-of-the-world-from-edge-to-core`.

Rowstron, A., Narayanan, D., Donnelly, A., O'Shea, G., and Douglas, A. (2012). Nobody ever got fired for using Hadoop on a cluster. In *Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing*, HotCDP '12, New York, NY, USA. ACM. doi: 10.1145 / 2169090. 2169092.

Sakalley, D. (2017). Using FPGAs to accelerate NVMe-of based storage networks. In *Flash Memory Summit 2017*, pages 8–11.

Sanaullah, A. and Herbordt, M. (2017). OpenCL for HPC/FPGAs: Case Study with 3D FFT. In *9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*, page 1–6. doi: 10.1145/3241793.3241800.

Sanaullah, A. and Herbordt, M. (2018a). An Empirically Guided Optimization Framework for FPGA OpenCL. In *2018 International Conference on Field Programmable Technology (FPT)*, pages 46–53. doi: 10.1109/FPT.2018.00018.

Sanaullah, A. and Herbordt, M. (2018b). FPGA HPC using OpenCL: Case Study in 3D FFT. In *9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*, page 1–6. doi: 10.1145/3241793.3241800.

Sanaullah, A. and Herbordt, M. (2018c). Unlocking Performance-Programmability by Penetrating the Intel FPGA OpenCL Toolflow. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*. doi: 10.1109/HPEC.2018.8547646.

Sanaullah, A., Khoshparvar, A., and Herbordt, M. (2016a). FPGA-Accelerated Particle-Grid Mapping. In *IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 192–195. doi: 10.1109/ FCCM .2016.53.

Sanaullah, A., Lewis, K., and Herbordt, M. (2016b). Accelerated Particle-Grid Mapping. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. `http://sc16.supercomputing.org/sc-archive/tech_poster/poster_files/post257s2-file3.pdf`.

Sanaullah, A., Lewis, K., and Herbordt, M. (2016c). GPU Accelerated Particle-Grid Mapping. In *IEEE High Performance Extreme Computing Conference*. DOI: 10.1109/ HPEC.2016.7761599.

Sanaullah, A., Sachdeva, V., and Herbordt, M. (2018a). SimBSP: Enabling RTL Simulation for Intel FPGA OpenCL Kernels. In *Proc. Heterogeneous High Performance Reconfigurable Computing*. doi: 10.1186/s12859-018-2505-7.

Sanaullah, A., Yang, C., Alexeev, Y., Yoshii, K., and Herbordt, M. (2018b). Real-Time Data Analysis for Medical Diagnosis using FPGA Accelerated Neural Networks. *BMC Bioinformatics*, 19 Supplement 18. doi: 10.1186/s12859-018-2505-7.

Schear, N., Cable, P. T., Moyer, T. M., Richard, B., and Rudd, R. (2016). Bootstrapping and maintaining trust in the cloud. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ACSAC '16, page 65–77, New York, NY, USA. Association for Computing Machinery. doi: 10.1145 / 2991079. 2991104.

Schonbein, W., Grant, R. E., Dosanjh, M. G. F., and Arnold, D. (2019). Inca: In-network compute assistance. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, New York, NY, USA. Association for Computing Machinery. doi: 10.1145 / 3295500. 3356153.

Shahzad, H., Sanaullah, A., and Herbordt, M. (2021). Survey and Future Trends for FPGA Cloud Architectures. In *IEEE High Performance Extreme Computing Conference*. DOI: 10.1109/HPEC49654.2021.9622807.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612 – 613. doi: 10.1145 / 359168. 359176.

Sheng, J., Humphries, B., Zhang, H., and Herbordt, M. (2014). Design of 3D FFTs with FPGA Clusters. In *IEEE High Performance Extreme Computing Conference*. doi: 10.1109/ HPEC.2014.7040997.

Sheng, J., Yang, C., Caulfield, A., Papamichael, M., and Herbordt, M. (2017). HPC on FPGA Clouds: 3D FFTs and Implications for Molecular Dynamics. In *27th International Conference on Field Programmable Logic and Applications*. doi: 10.23919/ FPL.2017.8056853.

Sheng, J., Yang, C., and Herbordt, M. (2015). Towards Low-Latency Communication on FPGA Clusters with 3D FFT Case Study. In *International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*. `https://pdfs.semanticscholar.org/832d/c69145f5ba0ed6a951583201b1b20dd2096e.pdf`.

Sheng, J., Yang, C., and Herbordt, M. (2016). Application-Aware Collective Communication on FPGA Clusters. In *IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. doi: 10.1109/ FCCM.2016.55.

Sheng, J., Yang, C., and Herbordt, M. (2018a). High Performance Dynamic Communication on Reconfigurable Clusters. In *28th International Conference on Field Programmable Logic and Applications*. doi: 10.1109/ FPL.2018.00044.

Sheng, J., Yang, C., and Herbordt, M. (2018b). High Performance Dynamic Communication on Reconfigurable Clusters (Extended Abstract). In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 219–219. doi: 10.1109/ FCCM.2018.00053.

Shumow, D. (2020). Incorrectly generated rsa keys: How to recover lost plaintexts. Cryptology ePrint Archive, Paper 2020/1059. `https://eprint.iacr.org/2020/1059`.

Smid, M. and Branstad, D. (1988). Data encryption standard: past and future. *Proceedings of the IEEE*, 76(5):550–559. doi: 10.1109 / 5.4441.

Songhori, E. M., Hussain, S. U., Sadeghi, A.-R., Schneider, T., and Koushanfar, F. (2015). Tinygarble: Highly compressed and scalable sequential garbled circuits. In *2015 IEEE Symposium on Security and Privacy*, pages 411–428. doi: 10.1109 / SP. 2015. 32.

Songhori, E. M., Riazi, M. S., Hussain, S. U., Sadeghi, A.-R., and Koushanfar, F. (2019). Arm2gc: Succinct garbled processor for secure computation. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA. Association for Computing Machinery. doi: 10.1145 / 3316781. 3317777.

Songhori, E. M., Schneider, T., Zeitouni, S., Sadeghi, A.-R., Dessouky, G., and Koushanfar, F. (2016). Garbledcpu: A mips processor for secure computation in hardware. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. doi: 10.1145 / 2897937. 2898027.

Stewart, L., Pascoe, C., Davis, E., Sherman, B., Herbordt, M., and Sachdeva, V. (2021). Particle Mesh Ewald for Molecular Dynamics in OpenCL on an FPGA Cluster. In *ArXiv Preprint arXiv:2009.12617*.

Sukhwani, B. and Herbordt, M. (2008). Acceleration of a Production Rigid Molecule Docking Code. In *2008 International Conference on Field Programmable Logic and Applications*, pages 341–346. doi: 10.1109/ FPL.2008.4629955.

Sukhwani, B. and Herbordt, M. (2010). FPGA Acceleration of Rigid Molecule Docking Codes. *IET Computers and Digital Techniques*, 4(3):184–195. doi: 10.1049/ iet-cdt.2009.0013.

Sukhwani, B. and Herbordt, M. (2014). Increasing Parallelism and Reducing Thread Contentions in Mapping Localized N-body Simulations to GPUs. In Kindratenko, V., editor, *Numerical Computations with GPUs*, pages 379–405. Springer Verlag. doi: 10.1007/ 978-3-319-06548-9_18.

Tehranipoor, M. and Koushanfar, F. (2010). A survey of hardware trojan taxonomy and detection. *IEEE Design & Test of Computers*, 27(1):10–25. doi: 10.1109 / MDT. 2010.7.

Tork, M., Maudlej, L., and Silberstein, M. (2020). *Lynx: A SmartNIC-Driven Accelerator-Centric Architecture for Network Servers*, page 117–131. Association for Computing Machinery, New York, NY, USA. doi: 10.1145 / 3373376. 3378528.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, page 1–11, New York, NY, USA. Association for Computing Machinery. doi: 10.1145 / 3338501. 3357370.

VanCourt, T. and Herbordt, M. (2004). Families of FPGA-based algorithms for approximate string matching. In *Proceedings. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2004.*, pages 354–364. doi: 10.1109/ ASAP.2004.1342484.

VanCourt, T. and Herbordt, M. (2005). LAMP: A tool suite for families of FPGA-based application accelerators. In *International Conference on Field Programmable Logic and Applications, 2005.* doi: 10.1109/ FPL.2005.1515797.

VanCourt, T. and Herbordt, M. (2006a). Application-dependent memory interleaving enables high performance in FPGA-based grid computations. In *IEEE Conference on Field Programmable Logic and Applications*, pages 395–401. doi: 10.1109/ FCCM.2006.25.

VanCourt, T. and Herbordt, M. (2006b). Sizing of processing arrays for FPGA-based computation. In *2006 International Conference on Field Programmable Logic and Applications*, pages 755–760. doi: 10.1109/ FPL.2006.311307.

VanCourt, T. and Herbordt, M. (2007). Families of FPGA-based accelerators for approximate string matching. *Microprocessors and Microsystems*, 31(2):135–145. doi: 10.1016/ j.micpro.2006.04.001.

VanCourt, T. and Herbordt, M. (2009). Elements of high performance reconfigurable computing. In Zelkowitz, M., editor, *Advances in Computers*, volume v75, pages 113–157. Elsevier. doi: 10.1016/ S0065-2458(08)00802-4.

VanCourt, T., Herbordt, M., and Barton, R. (2003). Case study of a functional genomics application for an FPGA-based coprocessor. In *International Conference on Field Programmable Logic and Applications*, pages 365–374. doi: 10.1016/ j.micpro. 2004.03.005.

VanRoekel, S. (2011). FedRAMP Policy Memo. `https://cic.gsa.gov/references/policy#FedRAMP`.

Vliegen, J., Rabbani, M. M., Conti, M., and Mentens, N. (2019). SACHa: self-attestation of configurable hardware. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 746–751. doi: 10.23919 / DATE. 2019. 8714775.

Wagh, S., Tople, S., Benhamouda, F., Kushilevitz, E., Mittal, P., and Rabin, T. (2020). Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229*.

Wang, T., Geng, T., Jin, X., and Herbordt, M. (2019a). Accelerating AP3M-Based Computational Astrophysics Simulations with Reconfigurable Clusters. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 181–184. doi: 10.1109/ ASAP.2019.000-5.

Wang, T., Geng, T., Jin, X., and Herbordt, M. (2019b). FP-AMR: A Reconfigurable Fabric Framework for Block-Structured Adaptive Mesh Refinement Applications. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 245–253. doi: 10.1109/ FCCM.2019. 00040.

Watson, J.-L., Wagh, S., and Popa, R. A. (2022). Piranha: A GPU platform for secure computation. Cryptology ePrint Archive, Paper 2022/892. `https://eprint.iacr.org/2022/892`.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469. doi: 10.1109 / TIFS. 2020. 2988575.

Will, M. A. and Ko, R. K. (2015). Chapter 5 - a guide to homomorphic encryption. In Ko, R. and Choo, K.-K. R., editors, *The Cloud Security Ecosystem*, pages 101–127. Syngress, Boston. `https://doi.org/10.1016/B978-0-12-801595-7.00005-7`.

Wolfe, P.-F., Patel, R., Munafo, R., Varia, M., and Herbordt, M. (2020). Secret Sharing MPC on FPGAs in the Datacenter. In *IEEE Conference on Field Programmable Logic and Applications*, pages 236–242. doi: 10.1109 / FPL50879. 2020. 00047.

Wu, C., Bandara, S., Geng, T., Sachdeva, V., Sherman, B., and Herbordt, M. (2021a). System-Level Modeling of GPU/FPGA Clusters for Molecular Dynamics Simulations. In *IEEE High Performance Extreme Computing Conference*. doi: 10.1109/HPEC49654.2021.9622838.

Wu, C., Geng, T., Bandara, S., Yang, C., Sachdeva, V., Sherman, W., and Herbordt, M. (2021b). Upgrade of FPGA Range-Limited Molecular Dynamics to Handle Hundreds of Processors. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. doi: 10.1109/ FCCM51124.2021.00024.

Wyden, R. (2019). Student Right to Know Before You Go Act of 2019. `https://www.wyden.senate.gov/download/summary-the-student-right-to-know-before-you-go-act-of-2019&download=1`.

Xiong, Q., Bangalore, P., Skjellum, A., and Herbordt, M. (2018a). MPI Derived Datatypes: Performance and Portability Issues. In *25th European MPI Users' Group Meeting*. doi: 10.1145/ 3236367.3236378.

Xiong, Q. and Herbordt, M. (2017). Bonded Force Computations on FPGAs. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 72–75. doi: 10.1109/ FCCM.2017.49.

Xiong, Q., Skjellum, A., and Herbordt, M. (2018b). Accelerating MPI Message Matching Through FPGA Offload. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 191–1914. doi: 10.1109/ FPL.2018.00039.

Xiong, Q., Yang, C., Haghi, P., Skjellum, A., and Herbordt, M. (2020). Accelerating mpi collectives with fpgas in the network and novel communicator support. In *IEEE Symposium on Field Programmable Custom Computing Machines*, pages 215–215. doi: 10.1109 / FCCM48280. 2020. 00046.

Xiong, Q., Yang, C., Patel, R., Geng, T., Skjellum, A., and Herbordt, M. (2019). GhostSZ: A Transparent SZ Lossy Compression Framework with FPGAs. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 258–266. doi: 10.1109/FCCM.2019.00042.

Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., and Ludwig, H. (2019). Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, page 13–23, New York, NY, USA. Association for Computing Machinery. doi: 10.1145 / 3338501. 3357371.

Yakoubov, S. (2017). A gentle introduction to yao's garbled circuits. `http://web.mit.edu/sonka89/www/papers/2017ygc.pdf`.

Yang, C., Geng, T., Wang, T., Patel, R., Xiong, Q., Sanaullah, A., Lin, C., Sachdeva, V., Sherman, W., and Herbordt, M. (2019a). Fully Integrated FPGA Molecular Dynamics Simulations. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1 – 31. doi: 10.1145/ 3295500.3356179.

Yang, C., Geng, T., Wang, T., Sheng, J., Lin, C., Sachdeva, V., Sherman, W., and Herbordt, M. (2019b). Molecular Dynamics Range-Limited Force Evaluation Optimized for FPGA. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 263–271. doi: 10.1109/ ASAP.2019.00016.

Yang, C., Sheng, J., Patel, R., Sanaullah, A., Sachdeva, V., and Herbordt, M. (2017). OpenCL for HPC with FPGAs: Case Study in Molecular Electrostatics. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. doi: 10.1109/ HPEC.2017. 8091078.

Yao, A. C. (1982). Protocols for Secure Computations. *Annual Symposium on Foundations of Computer Science - Proceedings*, pages 160–164. doi: 10.1109 / sfcs. 1982.38.

Yao, A. C. (1986). How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. doi: 10.1109 / SFCS. 1986.25.

Yarom, Y. and Falkner, K. (2014). FLUSH+RELOAD: A high resolution, low noise, l3 cache Side-Channel attack. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 719–732, San Diego, CA. USENIX Association.

Zahur, S., Rosulek, M., and Evans, D. (2015). Two halves make a whole. In Oswald, E. and Fischlin, M., editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 220–250, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zhang, C., Geng, T., Guo, A., Tian, J., Herbordt, M., Li, A., and Tao, D. (2022). H-GCN: A Graph Convolutional Network Accelerator on Versal ACAP Architecture. In *International Conference on Field-Programmable Logic and Applications*. DOI: TBD.

Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C.-Z., Li, H., and an Tan, Y. (2019). Secure multi-party computation: Theory, practice and applications. *Information Sciences*, 476:357–372. `https://doi.org/10.1016/j.ins.2018.10.024`.

Zhao, M. and Suh, G. E. (2018). FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244. doi: 10.1109 / SP.2018. 00049.

Zilberman, N., Audzevich, Y., Covington, G. A., and Moore, A. W. (2014). NetFPGA SUME: Toward 100 Gbps as Research Commodity. *IEEE Micro*, 34(5):32–41. doi: 10.1109 / MM.2014.61.

CURRICULUM VITAE