Exploring Hardware Accelerator Offload for the Internet of Things

Ryan A. Cooke, Suhaib A. Fahmy

Abstract: The Internet of Things is manifested through a large number of low-capability connected devices. This means that for many applications, computation must be offloaded to more capable platforms. While this has typically been cloud datacenters accessed over the Internet, this is not feasible for latency sensitive applications. In this paper we investigate the interplay between three factors that contribute to overall application latency when offloading computations in IoT applications. First, different platforms can reduce computation latency by differing amounts. Second, these platforms can be traditional server-based or emerging network-attached, which exhibit differing data ingestion latencies. Finally, where these platforms are deployed in the network has a significant impact on the network traversal latency. All these factors contributed to overall application latency, and hence the efficacy of computational offload. We show that network-attached acceleration scales better to further network locations and smaller base computation times that traditional server based approaches.

ACM CCS: Hardware → Hardware accelerators; Networks → Network services → Cloud computing; Networks → In-network processing

Keywords: Internet of Things, edge computing, hardware acceleration.

1 Introduction

Resource constrained Internet of Things (IoT) platforms at the network edge often need to offload compute intensive processing to more capable hardware in the cloud. While this can result in reduced computation time due to the more capable hardware available in cloud datacenters, it costs additional communication latency due to the need for data to travel to the cloud and the response to return to the IoT node. Hence, the application acceleration factor– the relative reduction in computation latency– must be enough to overcome this additional communication latency penalty. Edge computing is a broad paradigm in which processing is moved from high performance centralised resources towards the data source at the periphery of the network. Computing resources are typically less capable at edge nodes, but communication is minimised since data need not be moved up the network to be computed on. With the Internet of Things driving an explosive growth in connectivity of resource constrained computing platforms, we must consider how to address the latency implications of cloud-based computing offload.

Latency is a key performance metric in a variety of applications. In augmented or virtual reality applications, user experience is directly tied to latency. Industrial automation systems have stringent latency requirements that have productivity and safety implications, as do smart vehicles. These scenarios often include computationally complex operations such as object detection and classification, which can be too challenging for resource constrained nodes to perform. When offloading, the total latency of these applications is determined by the sum of the computation latency, and the communication latency required to reach the offload platform.

In order to reduce computation latency, more capable hardware is often exploited in offload platforms and this is what enables the trade-off against increased communication latency. The cloud hosts powerful server class processors that are much more capable than the microcontrollers or single board computers typically available in IoT nodes. Servers can be scaled and computing performance increased depending on application requirements. There has also been an increasing trend of adopting heterogeneous specialised hardware in the datacenter to further improve performance and efficiency. FPGAs have recently seen increased use in the datacenter due to their flexibility and increased performance per watt compared to CPUs and GPUs in various applications [1, 2, 3, 4]. As latency requirements have increased in importance and hardware at the network edge has improved, the benefits of offloading to centralised computing resources is now heavily impacted by inherent communication delays.

Reducing communication latency has been tackled through moving the more capable processing platforms closer to the edge. Cloudlets are small-scale datacenters or servers deployed close to data sources, in an attempt to provide cloud-like services a few hops away in the network [5, 6, 7]. Data now traverses a few switches over a LAN instead of the Internet, reducing communication delay and improving predictability. They have been shown to reduce latency in a range of applications, providing greater computing capabilities to less capable edge nodes. Cloudlets may utilise capable hardware comparable to that found in larger cloud datacenters, including hardware accelerators. While network communication latencies are reduced, these platforms are based on traditional server architectures are still subject to typical latency penalties from the software network stack, PCIe interconnect, and competition for resources between applications.

While moving processing closer to the edge reduces the communication latency by reducing the time taken for data to traverse the network, a significant proportion of the latency can be attributed to the time taken for data to reach the computational resources once it arrives at the target offload platform. Network interface cards, PCI Express interconnect, and the software network stack, among other factors, all contribute to this non-deterministic delay [8]. Accelerators in the datacenter typically augment compute servers over PCI Express, entailing further latency to exploit their acceleration capabilities [9]. In [10], these ingestion latencies were identified to be key contributors to the latency of a DNN application when hardware acceleration reduces computation latency sufficiently. As hardware improvements reduce the computation latency, and moving processing closer to the edge reduces the network traversal time, network data ingestion latency becomes an important factor, and a potential bottleneck for IoT applications with offloaded computation.

Network ingestion latencies can be reduced through tighter coupling of hardware acceleration to the network interface. FPGAs offer a highly capable acceleration platform as they allow the design and implementation of customised datapaths tailored to specific applications, thereby offering efficient high performance computing. An additional benefit is that they can be tightly coupled to network interfaces, to receive data directly from the network without it having to traverse PCIe or a software network stack in a host server, thereby significantly improving network ingestion latency. FPGAs can be integrated into network interface cards (NICs), switches, or IoT gateways to process data before it reaches a server. This tight interface coupling has significant advantages for lightweight offload nodes, where standard ingestion latencies can be high.

In this paper, we explore the trade-off between computation acceleration and offload platform location, with consideration for network ingestion latency. We demonstrate that as processing is moved closer to the edge, ingestion latency becomes an increasingly important component of total application response time. We also examine the effect of reducing the ingestion latency through network attached acceleration platforms and show how applications with different computation to communication latency ratios are affected.

2 Related Work

For data to move to a large cloud platform such as Amazon EC2 it must travel through a WAN such as the internet, resulting in large and non-deterministic delays, in the range of 10–100s of milliseconds depending on datacenter location [11]. Edge computing is a broad term that encompasses various methods of implementing data processing closer to sources. This can often reduce latency as data does not have to travel through as large of a network to reach the application. In this section we will discuss related work in these areas.

2.1 Cloudlet Servers

Cloudlets provide locally accessible increased computing capability to less capable nodes, often over a LAN. They act as small scale cloud services, with lower latency response times in comparison to larger datacenter platforms [12]. One growing application of cloudlets is for compute-intensive mobile applications, where offloading parts of the application can result in improvements to both latency and energy consumption of the user device [13, 14]. Image processing applications are often utilised to demonstrate the effectiveness of these platforms. Applications such as face recognition [5, 6], augmented reality [7] and video surviellance [15] have all been demonstrated on cloudlet platforms with significant latency improvements over traditional cloud offloading. These cloudlets are attractive for application that are time sensitive, but where the computation is too intensive to be done on less capable hardware at data sources.

Datacenters – and thus the similar but smaller scale cloudlets – have seen an increased deployment of hardware accelerator devices in recent years. Platforms such as Graphics Processing Units (GPUs) and FPGAs are augmenting traditional CPUs as their performance scaling has slowed. Compute intensive functions can be offloaded to these accelerators over the PCIe interconnect in a server, and computation latency reduced through the use of hardware structures optimised for a specific set of applications. FPGAs in particular are seeing interest due to their flexibility, and performance per watt compared to CPUs and GPUs

[16, 17, 18, 19] and their virtualisation support to allow sharing among multiple applications, and dynamic adaptation to workloads [20].

While cloudlets are in closer proximity to the data sources, and thus reduce communication latency over the network, there are multiple sources of delay within the cloudlet that can increase the total latency of the application, as with full-scale datacenters [21, 22, 23]. Virtualisation, multi-tenant contention, and PCIe offload [9] all add delay and variability.

2.2 Edge Node Computing

We consider the term edge node computing to refer to computing performed at the data source, or at a node directly connected to it, such as a cluster head or IoT gateway. Resources at these nodes are generally less capable than at a cloudlet, but there is little communication latency. Often, micro controllers or single board computers running embedded Linux are used in these deployments, with smaller microcontrollers running real time operating systems for sensor nodes. Smart gateways are often used for IoT applications, collecting data from sensor nodes through WiFi or Bluetooth and performing processing. This has seen use in medical applications and manufacturing [24, 25, 26], where performing processing at a gateway node results in lower latency and energy consumption than doing it at the sensor nodes.

FPGAs have also seen use for edge node acceleration, utilizing SoC platforms such as the Xilinx Zynq that tightly couple an FPGA with an Arm processor. This has seen use in gas sensor networks and industrial applications [27, 28], and involves software running on the processor system controlling and communicating with the FPGA fabric. These edge devices typically house smaller FPGAs with fewer resources, meaning that certain optimisations of the accelerator must be sacrificed in order to meet resource constraints.

2.3 In-Network Computing

Network infrastructure such as switches, bridges, and larger gateways connect edge nodes to each other, cloudlets, and to wider networks such as the Internet [29]. In-network computing utilises these intermediate nodes to perform processing of data, and since these nodes are in closer proximity to data sources and are computationally capable, they are not subject to many of the latencies inherent in cloudlet or datacenter offload.

FPGAs are an attractive platform for in-network processing due to their flexibility, high performance processing, and they are often already present within the network infrastructure, used for packet processing and network services. Studies have shown how their low latency connectivity can enable singificantly improved computing offload in a network-attached setup [30]. This approach has recently seen use within datacenter applications [31, 18], and has been shown to improve latency compared to traditional techniques. FPGA based systems have been used to implement smart-switches or smart-NICs, which perform data processing in addition to their usual network functions. Unlike FPGA accelerators used in server based systems, these devices have minimal software involvement, and do not rely on PCIe to transfer data from the network port to the acceleration fabric.

3 Scenario

In [10], we compared computation latency and ingestion latency for a deep neural network (DNN) image classification application using a variety of accelerated computing platforms. That study demonstrated that ingestion latency can have a significant impact on the overall latency reduction achievable for different platforms. Images generated at a Raspberry Pi edge node were transmitted to different offload platforms over a network. Performing the DNN computation on the edge node itself resulted in a computing latency of 2.3s due to the constrained capabilities of the embedded processor on that node. Offload targets investigated included server based platforms where data enters the system through a PCIe network card and is moved to an accelerator PCIe card via a controlling application running in Linux userspace. This is representative of a typical host deployment in a cloudlet or datacenter. The same experiments were also explored with network attached FPGA accelerators integrated into a network switch. In this deployment, packets received at the network interface of the switch were forwarded to the accelerator implemented on the same FPGA fabric depending on pre-specified packet headers. From these experiments,

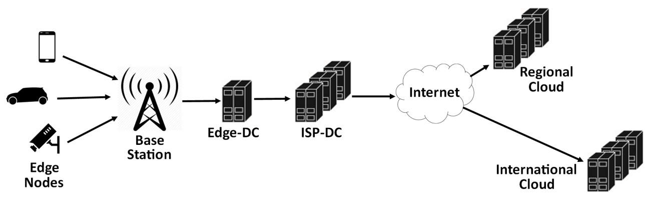

Figure 1: Network scenario used for the discussion in this paper. It is typical of an Internet of Things deployment.

we measured an average ingestion latency of 60ms for the entire image for a server based platform, and 1ms for the network attached FPGA platform. While both platforms resulted in the same reduction in computing latency, down to 60ms, the low ingestion latency of the network attached platform gave it superior overall latency.

In [32], the authors carried out a series of experiments measuring the network traversal time of packets from a mobile phone source to various locations that could be used to offload processing. This includes the eNodeB base station, a telco central office re-architected as a datacenter, the ISP datacenter, and various Amazon Web Services (AWS) virtual machines (VMs) in different geographic locations. These different offload locations have varying network distances to the data sources and hence result in a range of different network latencies, as reproduced in Table 1.

Table 1: Network traversal time to various offload locations, measured in [32].

Location	Latency (ms)
BS – Base station	28
EdDC – Edge datacenter	41
ISPDC – ISP datacenter	62
RgCld – Regional cloud	77
IntCld – International cloud	151

The overall network scenario is typical of an Internet of Things deployment, and can be seen in Figure 1: data is generated at the edge node and transmitted to the nearest access point, in this case, a base station a single hop away. From there, it travels through multiple network hops to a local edge datacenter, then to the ISP datacenter, and finally to a cloud datacenter, which can be located anywhere across the Internet. With each successive hop, additional network latency is introduced. Once the data reaches the target destination, it must be ingested by the computing platform and processed, with the result sent back to the edge node source. The time taken to complete this process is the total application latency.

Each of these networked locations could potentially host computing platforms to perform this computation. More capable accelerator hardware can also be deployed to be shared across multiple edge node clients and perform the computation with lower latency. Typically, with each successive hop, there is an increase in available computing resources and therefore opportunities to reduce computation latency further. Each location can host a range of computing platforms, which can reduce computation latency by varying amounts. For example, in [10], performing the computation on a server in software as opposed to the edge node reduced computation latency by 7×, and performing it on an FPGA accelerator attached to the server improved computation time by almost 40× compared to the edge node and almost 6× compared to the server.

We want to examine the relationship and trade-offs present between the amount the offload platform – such as a server or hardware accelerator – reduces the computation time, and the communication latency due to moving data to the that platform. Using the results previously discussed we estimate total offload latency for similar streaming applications offloaded onto different platforms deployed at these different network locations, while considering varying acceleration factors achievable for different platforms. We represent the degree to which the computation latency is reduced by the offload platform using an acceleration factor, such that:

$$\text{acceleration factor} = \frac{\text{latency}_{\text{comp}_{\text{base}}}}{\text{latency}_{\text{comp}_{\text{offload}}}} \tag{1}$$

where the base computation latency is the latency when performing processing at the IoT edge node. The acceleration factor is dependant on the platform being used to carry out the computation, such as a server class processor, or hardware accelerator like an FPGA, not on the location where the platform is hosted.

The total application latency hence depends on that computation time, the latency for data to be sent to the offload platform, and the ingestion latency at that platform, estimated as:

$$\begin{aligned} \text{latency}_{\text{total}} = \text{latency}_{\text{comp}} &+ \text{latency}_{\text{ingestion}} \\ &+ \text{latency}_{\text{network}} \end{aligned} \tag{2}$$

While this model does not capture all the details of a real implementation, such as network congestion, it is detailed enough to allow us to reason about the mix of computation offload platforms and where to locate them for improved application latency.

4 Discussion

The results for total latency estimation for each location and platform can be seen in Figure 2. Intuitively, as the acceleration factor increases, the computation time decreases, reducing total latency. However this provides diminishing returns, as the communication latency begins to dominate as computation latency reduces. This means that communication latency limits achievable performance when the computational complexity results in comparable computation latencies. Beyond a certain point, further increasing the processing capability of the offload platform results in minimal overall latency improvement improvement.

Utilising network attached accelerator platforms, shown with the dashed lines, reduces total latency further for a given location, due to reduction of ingestion latency. The communication latency required to reach platforms further from the edge means less computationally capable platforms closer to the edge can sometimes provide better performance overall than more capable platforms further away. Similarly, utilising network-attached computing platforms further away can result in lower latency than a more local platform using server based compute, since ingestion latency is significantly reduced. We can see that a strategy of increasing computing capability has limits in terms of achievable latency and that reduction of communication latency is ultimately required to improve overall latency further.

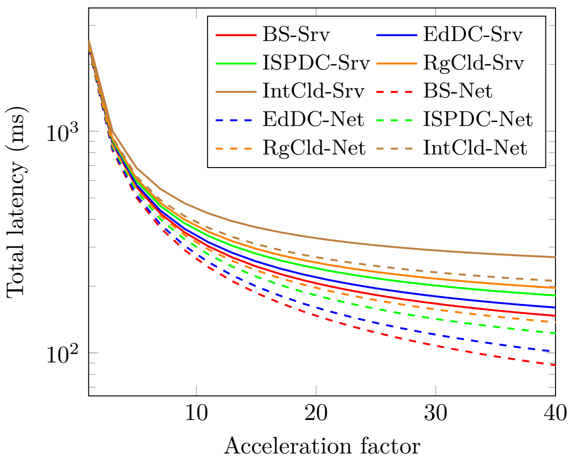

Figure 2: Estimated total latencies when computing is offloaded to the network locations in Table 1, for varying acceleration factors. Solid lines represent server based acceleration platforms, and dashed lines represent network-attached acceleration platforms.

The closer processing is moved to the edge, the greater the relative improvement achieved through network-attached processing. The same can be seen as acceleration factor increases – the relative difference between the server and server-less deployments increases in turn. Figure 3 demonstrates this further, showing the relative reduction in total latency achieved when using network-attached over server-based acceleration at each location, and similarly shows that network attached platforms yield greater benefits when applied to platforms closer to the edge, and for higher acceleration factors. This is because as the computation and network traversal times reduce, through improved processing capabilities and moving the compute closer to the source, the ingestion latency becomes a relatively more significant contributor to overall latency. For example, for an acceleration factor of 40×, for a base station offload (BS), using server based computing, total latency was around 140ms. Ingestion latency contributed 60ms to this total, over 40%, greater than each of the network traversal and computation latency. By comparison, using a network-attached accelerator resulted in a total latency of 85ms, and ingestion latency contributed only 1%.

These results are based on the characteristics of the DNN application used in [10], which had a base computation latency of 2.3s on the Raspberry Pi edge node. In this situation, the computation latency by far outweighs the communication latency, so benefits can be achieved without requiring a significant acceleration factor on the offload platform. We now explore how this analysis changes when the balance of computation latency and communication latency changes. Figure 4 shows the results when the base computation latency is 200ms as opposed to 2.3s, and hence closer to the magnitude

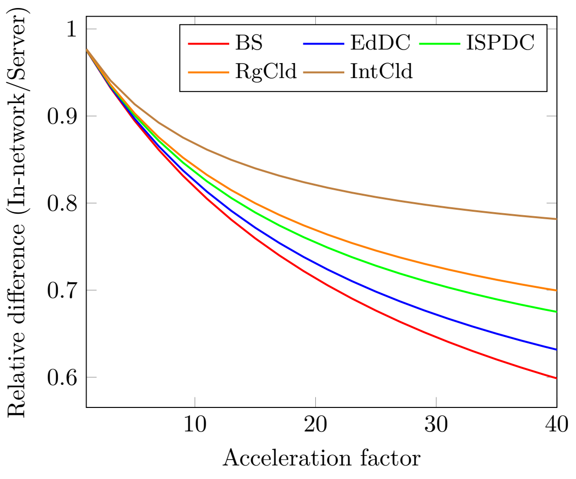

Figure 3: Relative reduction of total latency provided by network-attached accleration compared to server-based, when computing is offloaded to the different networked locations in Table 1.

of the network latency. In this scenario, we can see that depending on the location and the ingestion latency of the platform, a greater acceleration factor is required to justify offloading. The base station (BS) must be able to perform processing at least 3× as fast as the edge node if using a server based platform, but only around 1.4× faster when using network-attached acceleration. In this situation the furthest AWS instance as a server based accelerator can never achieve an improvement over the edge node, even for extremely large acceleration factors, though network-attached acceleration would still be feasible.

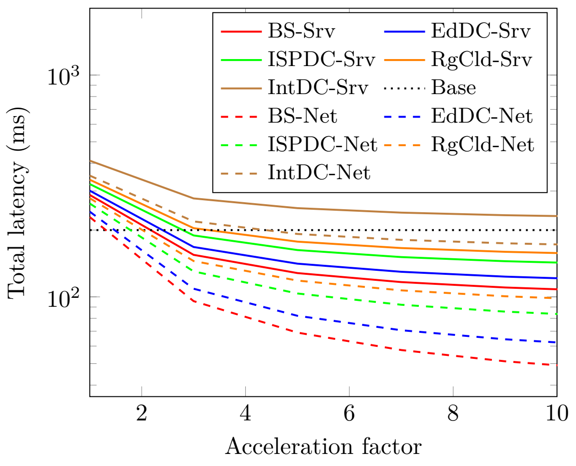

Figure 4: Estimated total latencies when a 200ms base computation time is offloaded to the different networked locations in Table 1, for varying acceleration factors. Solid lines represent server based acceleration, and dashed lines network attached. The black dotted line shows the base computing latency.

When the base computation time is reduced further, to 100ms, an even greater acceleration factor is needed to justify offload, as shown in Figure 5. Even at the base station, the resource closest to the edge node, achieving increased latency performance would be a challenge unless using network attached processing. Any of the other platforms wouldn't achieve improvements without utilising this, regardless of acceleration factor.

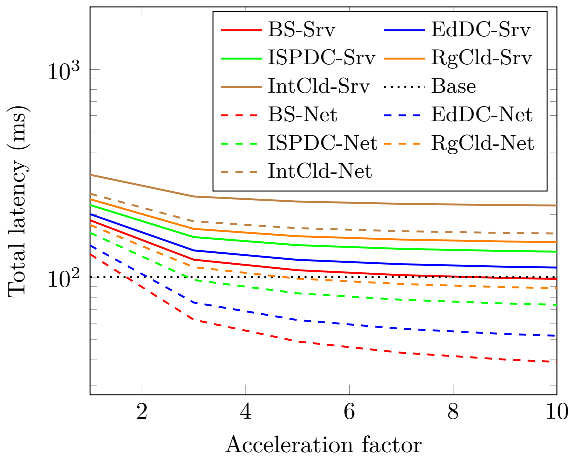

Figure 5: Estimated total latencies when a 100ms base computation time is offloaded to the different networked locations in Table 1, for varying acceleration factors. Solid lines represent server based acceleration, and dashed lines network attached. The black dotted line shows the base computing latency.

This study shows that deploying network-attached acceleration offers new opportunities to offload smaller IoT tasks that may have traditionally not have benefited from offloading. Additionally, more lightweight accelerators can be deployed that require less computing power, while still achieving reductions in total latency. These network-attached accelerator platforms also scale better to servicing multiple IoT edge nodes, as shown in [10]. One factor not considered in this study is the cost of deploying hardware accelerators to reduce latency. This analysis can be carried out in further work. The mathematical model detailed in [30] can also be used to model cost.

5 Conclusion

Computational offload is an attractive method of reducing processing time for latency sensitive applications. While computation time is reduced, there is an associated communication latency caused by the traversal to the offload location, and the time taken to move the data to the processing platform at that location. Computational resources can be increased through the use of more powerful hardware, or using specialised accelerators. We presented a case study for an image processing application where an edge node offloaded processing to one of several locations, ranging from a base station a single hop away, to a cloud datacenter in another continent. This demonstrated that the communication cost limits the achievable total latency reduction when increasing the computational resources available. While efforts can be made to improve hardware and bring down processing time, this offers diminishing returns.

Reducing the communication time through moving the processing closer to the data source can reduce the total latency. However this only effects the time taken for data to traverse the network to the offload location. Our case study showed that the ingestion latency, the time taken for data to traverse the network interfaces and software stacks of the compute platforms, is a considerable contributor to the overall latency. Utilising network attached processing platforms that bypass these interfaces reduces ingestion latency, and thus allows greater potential to reduce the total latency through both moving compute to the edge, and increasing hardware capability. As we improve computation, and reduce network traversal, the next step to increasing performance is tackling how data is managed consumed by the compute.

This opens up opportunities to offload smaller tasks that may have traditionally not have benefited from computational offload. Likewise, it could enable the use of more lightweight processing that requires less hardware resources, performing more, with less – as well as open up hardware for sharing across multiple applications or client devices.

Acknowledgement

This work was supported in part by The Alan Turing Institute under the UK EPSRC grant EP/N510129/1.

Literature

[1] H. M. Hussain, K. Benkrid, A. T. Erdogan, and H. Seker, "Highly parameterized k-means clustering on fpgas: Comparative results with GPPs and GPUs," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs*, no. 1, 2011, pp. 475–480.

[2] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA accelerators for efficient cloud computing," in *Proceedings of the International Conference on Cloud Computing Technology and Science (CloudCom*, 2015, pp. 430–435.

[3] Y. R. Qu, H. H. Zhang, S. Zhou, and V. K. Prasanna, "Optimizing many-field packet classification on FPGA, multi-core general purpose processor, and GPU," in *ANCS*, 2015.

[4] A. Fiessler, S. Hager, B. Scheuermann, and A. W. Moore, "HyPaFilter: a versatile hybrid FPGA packet filter," in *ANCS*, 2016.

[5] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," 2012, pp. 000 059–000 066.

[6] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *HotWeb*, 2016, pp. 73–78.

[7] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *MobiSys '14*, 2014, pp. 68–81.

[8] N. Zilberman, M. Grosvenor, N. Manihatty-bojan, D. A. Popescu, G. Antichi, S. Galea, A. Moore, R. Watson, and M. Wojcik, "Where has my time gone?" in *International Conference on Passive and Active network measurement*, 2017.

[9] R. Neugebauer, G. Antichi, J. F. Zazo, S. López-buedo, and A. W. Moore, “Understanding PCIe performance for end host networking,” in *SIGCOMM*, 2018, pp. 327–341.

[10] R. A. Cooke and S. A. Fahmy, “Quantifying the latency benefits of near-edge and in-network FPGA acceleration,” in *Proceedings of the International Workshop on Edge Systems, Analytics and Networking (EdgeSys)*, 2020, pp. 7–12.

[11] O. Tomanek, P. Mulinka, and L. Kencl, “Multidimensional cloud latency monitoring and evaluation,” *Computer Networks*, vol. 107, no. Part 1, pp. 104–120, 2016.

[12] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, “The case for VM-based cloudlets in mobile computing,” *Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[13] Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, “Resource effiicient mobile computing using cloudlet infrastructure,” in *Conference on Mobile Ad-hoc and Sensor Networks*, 2013.

[14] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, “Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing,” *Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[15] M. Satyanarayanan, P. B. Gibbons, L. Mummert, P. Pillai, P. Simoens, and R. Sukthankar, “Cloudlet-based just-in-time indexing of IoT video,” in *GIoTS 2017*, 2017.

[16] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, “Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?” in *International Symposium on Field-Programmable Custom Computing Machines*, 2012, pp. 232–239.

[17] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J. Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, “A reconfigurable fabric for accelerating large-scale datacenter services,” *IEEE Micro*, vol. 35, no. 3, pp. 10–22, 2015.

[18] “Azure accelerated networking : SmartNICs in the public cloud,” in *NSDI*, 2018, pp. 51–64.

[19] A. M. Caulfield, E. S. Chung, P. Kaur, J.-y. K. Daniel, L. Todd, and M. Kalin, “A cloud-scale acceleration architecture,” in *MICRO*, 2016.

[20] M. Asiatici, N. George, K. Vipin, S. A. Fahmy, and P. Ienne, “Virtualized Execution Runtime for FPGA Accelerators in the Cloud,” *IEEE Access*, vol. 5, pp. 1900–1910, 2017.

[21] J. Whiteaker, F. Schneider, and R. Teixeira, “Explaining packet delays under virtualization,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 39–44, 2011.

[22] R. Shea, F. Wang, H. Wang, and J. Liu, “A deep investigation into network performance in virtual machine based cloud environments,” in *INFOCOM*, 2014, pp. 1285–1293.

[23] L. Chen, S. Patel, H. Shen, and Z. Zhou, “Profiling and understanding virtualization overhead in cloud,” in *International Conference on Parallel Processing*, vol. 2015-Decem, 2015, pp. 31–40.

[24] Q. Qi and F. Tao, “A smart manufacturing service system based on edge computing, fog computing and cloud computing,” *IEEE Access*, vol. 7, 2019.

[25] C.-H. Chen, M.-Y. Lin, and C.-C. Liu, “Edge computing gateway of the industrial internet of things using multiple collaborative microcontrollers,” *IEEE Network*, vol. 32, pp. 24–32, 2018.

[26] A. M. Rahmanu, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, “Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach,” *FGCS*, vol. 78, pp. 641–658, 2017.

[27] X. Zhai, A. A. S. Ali, A. Amira, and F. Bensaali, “MLP neural network based gas classification system on Zynq SoC,” *IEEE Access*, vol. 4, pp. 8138–8146, 2016.

[28] M. Urbina, A. Astarloa, J. Lazaro, U. Bidarte, I. Villalta, and M. Rodriguez, “Cyber-physical production system gateway based on a programmable SoC platform,” *IEEE Access*, vol. 5, 2019.

[29] Cisco, “The Cisco edge analytics fabric system,” 2016.

[30] R. A. Cooke and S. A. Fahmy, “A model for distributed in-network and near-edge computing with heterogeneous hardware,” *Future Generation Computer Systems*, vol. 105, pp. 395–409, 2020.

[31] Y. Tokusashi, H. T. Dang, F. Pedone, R. Soulé, and N. Zilberman, “The case for in-network computing on demand,” in *Proceedings of the EuroSys Conference*, 2019, pp. 1–16.

[32] A. Cartas *et al.*, “A reality check on inference at mobile networks edge,” in *Proceedings of the International Workshop on Edge Systems, Analytics and Networking (EdgeSys)*, 2019, pp. 54–59.

Mr. Ryan A. Cooke is a PhD student in the School of Engineering at the University of Warwick, UK, where he also received his M.Eng. degree in Electronic Engineering in 2015.
His research interests include reconfigurable computing, and in-network analytics acceleration.

Address: University of Warwick, School of Engineering, Library Road, Coventry, CV4 7AL, United Kingdom, E-Mail: ryan.cooke@warwick.ac.uk

Dr. Suhaib A. Fahmy is Reader in Computer Engineering at the University of Warwick, where his research encompasses reconfigurable computing, high-level system design, and computational acceleration of complex algorithms.
He received the M.Eng. degree in information systems engineering and the Ph.D. degree in electrical and electronic engineering from Imperial College London, UK, in 2003 and 2007, respectively. From 2007 to 2009, he was a Research Fellow with Trinity College Dublin and a Visiting Research Engineer with Xilinx Research Labs, Dublin. From 2009 to 2015, he was an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore.
Dr. Fahmy was a recipient of the Best Paper Award at the IEEE Conference on Field Programmable Technology in 2012, the IBM Faculty Award in 2013 and 2017, the Community Award at the International Conference on Field Programmable Logic and Applications, the ACM TODAES Best Paper Award in 2019 and is a senior member of the IEEE and ACM.

Address: University of Warwick, School of Engineering, Library Road, Coventry, CV4 7AL, United Kingdom