SmartNICs at Edge for Transient Compute Elasticity

Diman Zad Tootaghaj[†], Anu Mercian[★], Vivek Adarsh[**], Mehrnaz Sharifian[*‡], Puneet Sharma[†]
[†]Hewlett Packard Labs, [★]LBNL, [*]Aruba Networks, [‡] UC Davis,[**] UC Santa Barbara

ABSTRACT

This paper proposes a new architecture that strategically harvests the untapped compute capacity of the SmartNICs to offload transient microservices workload spikes, thereby reducing the SLA violations while providing better performance/energy consumption. This is particularly important for ML workloads at Edge deployments with stringent SLA requirements. Usage of the untapped compute capacity is more favorable than deploying extra servers, as SmartNICs are economically and operationally more desirable. We propose SpikeOffload, a low-cost and scalable platform that leverages machine learning to predict the spikes and orchestrates seamless offloading of generic microservices workloads to the SmartNICs, eliminating the need for pre-deploying expensive host servers and their under-utilization. Our SpikeOffload evaluation shows that SLA violations can be reduced by up to 20% for specific workloads. Furthermore, we demonstrate that for specific workloads our approach can potentially reduce capital expenditure (CAPEX) by more than 40%. Also, performance per unit energy consumption can be improved by upto 2×.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Networks →** *Network performance analysis*; *Programming interfaces*.

KEYWORDS

SmartNIC, Edge, Application Offload, Serverless Computing

ACM Reference Format:
Diman Zad Tootaghaj[†], Anu Mercian[★], Vivek Adarsh[**], Mehrnaz Sharifian[*‡], Puneet Sharma[†]. 2022. SmartNICs at Edge for Transient Compute Elasticity. In *DistributedML (DistributedML '22), December 9, 2022, Roma, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3565010.3569065

1 INTRODUCTION

The growth of edge-computing systems is being driven by machine learning solutions and applications requiring high performance and low latency, such as Internet-of-Things (IoT), manufacturing, and other operational technologies. Furthermore, edge applications are increasingly adopting microservices frameworks, including serverless. However, provisioning at the edge for additional computing needs on an ephemeral basis for sudden workload spikes, or as we call it *transient elasticity*, is non-trivial [1, 2], especially for ML inferencing workloads. Scaling microservice applications at the edge poses critical challenges. Service-level agreement (SLA) violations typically occur due to workload bursts leading to short-term compute power shortages on the edge servers [3]. Since SLA violations for ML workloads carry severe penalties, one common way to eliminate violations is to over-allocate resources preemptively. This leads to long refresh cycles and under-utilization of expensive resources. Meanwhile, SmartNICs (smart network interface cards) are gaining popularity. They are being increasingly deployed for offloading various network functions and provide real-time line rate at scale [4, 5]. However, the additional general-purpose compute capacity available on these SmartNICs has largely been untapped. We propose SpikeOffload, a novel Edge-Computing platform that leverages heterogeneous-computing nodes (including domain-specific accelerators (DSA) like SmartNICs) to acquire appropriate computation resources to handle the workload and transient spikes. We predict the possibility of incoming workload spikes and intelligently load-balance containerized workloads across the heterogeneous-compute resources, including untapped general-purpose, compute on SmartNICs when demand escalation is imminent. We demonstrate SpikeOffload using Serverless workloads (more in Sec. 2.3) as they are low compute-intensive workloads that are ideal for limited resource DSAs like SmartNICs. SmartNICs are desirable candidates for serverless application offload because (1) they are closer to the data ingress-side and bypass the network stack overhead, (2) there is availability and proximity of SoC-based general-purpose compute to the server for application processing [6, 7], (3) from our analysis, we observe close to 30% unused compute cycles on the SmartNICs that can be re-purposed for non-networking workloads, and (4) they eliminate the costly need to provision traditional servers for transient compute requirements. To the best of our knowledge, we are the first to offload containerized non-networking workloads such as ML inferencing to SoC-based SmartNICs.

Although prior works have studied the applicability of offloading specific parts of applications, e.g., using P4 programmability, actor-programming paradigm, etc., [8–10], these works are limited to specific applications and require code modification for other types of workload/application offloading. However, our work, by virtue of offloading the entire container into the SmartNIC, is application-agnostic and can be used for any containerized application. As we describe later in the paper, such offloading is of particular interest for ML workloads with strict SLA requirements at the Edge.

SpikeOffload's orchestrator can be generalized for scaling edge across multiple servers and different kinds of SmartNICs. Stated otherwise, our approach can scale application offload across different dimensions of heterogeneity. This approach enables us to secure a competitive advantage compared to current homogeneous edge architectures and deployments. To this end, this paper makes the following **main** contributions:

Some authors of this paper have since moved, and their affiliations have changed. All their contributions to this paper were done while they were HPE employees.

DistributedML '22, December 9, 2022, Roma, Italy
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9922-7/22/12…$15.00
https://doi.org/10.1145/3565010.3569065

- Designing a novel architecture that leverages heterogeneous computing nodes (SmartNICs and Host server) to facilitate efficient handling of spikes in processing demand at the edge;
- Development and characterization of workload predictor and orchestrator that work in tandem to reduce SLA violations while efficiently handling workload spikes;
- Characterization of competitive advantages of our architecture through an in-depth analysis of capital expense costs and overhead savings from reducing SLA violations. Our examination demonstrates that capital expenditure can be reduced by more than 40%, while performance/energy consumption can be tuned up by a factor of 2. In addition, our architecture reduces SLA violations by as much as 20%.

2 BACKGROUND AND MOTIVATION

In this section, we provide the background and motivation for SpikeOffload, with an overview of multicore SoC-based SmartNICs, and the need for DSA such as SmartNICs in the Edge Computing platform. In addition, we also briefly discuss an overview of the Serverless Platform and how they integrate into edge computing use-cases such as ML inference.

2.1 Network Accelerators: SmartNICs

SmartNICs are domain-specific accelerators, specifically advantageous for Networking functionalities. There are broadly three categories of network accelerators or SmartNICS: ASIC, FPGA, and SoC-based SmartNICs [9, 11]. In this study, we only focus on SoC-based SmartNICs. Multicore SoC-based SmartNICs use embedded CPU cores to process packets, trading performance to provide substantially better programmability than ASIC-based designs. (e.g., DPDK-style code can be directly run on a familiar Linux environment). For instance, Mellanox Bluefield [6] uses general-purpose CPU cores (ARM64), while others, like Netronome [12], have specific cores for network processing.

SoC-based SmartNICs (e.g., Mellanox) have two modes of operation: *Embedded*, and *Separated* modes. The interfaces are mapped to the host OS network stack in embedded mode, and the kernel routes packets from the host. The host OS and the SmartNIC have separate, independent network stacks to process packets in the separated mode. While we observe slightly better tail-latencies from packet processing in embedded mode, the offset from separate mode is negligible. For SpikeOffload, we adopt the separated mode due to its programmable flexibility and the ability to run containers directly on the SmartNIC's ARM64 OS. We observe that the recent versions of SmartNICs can support containers with many network restrictions. Our hope is SpikeOffload, can work with both modes with more improvements from the vendor in the underlying hardware or by providing the compatible container network interface (CNI).

2.2 Need for Accelerators at the Edge

Recently, there has been much interest in using SmartNICs in cloud data center servers to boost performance by offloading computation in servers by enhancing network datapath processing. This section explains why SmartNICs are essential in the new generation of high-performance edge computing servers. The cost of building an interconnection network for a large edge cluster with cloud support can significantly affect the choice of design decisions. With increasing network interface bandwidths, the gap between the network performance and compute performance is widening. This has resulted in increased adoption and deployment of SmartNICs [5]. If SmartNICs were leveraged to offload only network functionalities, it would add 30% more computational capacity to the current servers [13]. Typically, SoC-based SmartNICs are generally priced at 25-30% the cost of high-performance edge Servers. Therefore, adding a SmartNIC to perform only network functions is a wise decision. However, the SmartNICs can do more than network functions. The compute capacity of an SoC-based SmartNIC is generally around 40-50% of server compute capacity as per our initial analysis. If additional compute is required within this range, exploiting the compute capacity of SmartNICs to manage workload spikes instead of servers is a more economical decision. Currently, all that compute that is available on SmartNICs is primarily used for offloading network functions and services. In most cases, that is a severe under-utilization of the available compute power on SmartNICs. We aim to harvest this under-utilized compute and make it available to the applications.We propose a novel platform to orchestrate general purpose compute workloads onto deployed SmartNICS.

2.3 Serverless Edge Computing

Serverless computing, or *Function as a Service* is emerging as a new and compelling computing paradigm where applications can be deployed as multiple lightweight tasks in the form of functions on a shared runtime environment. Serverless has been popularly adopted by public cloud vendors such as Amazon, Microsoft, and Google [14, 15]. The flexible low-compute intensive feature makes Serverless computing very lucrative for edge computing, as application workloads can be easily balanced between edge and cloud. Moreover, since the serverless workload usually consists of lightweight functions, it provides an excellent opportunity for SmartNICs to accelerate them. The elasticity feature in serverless computing allows users to acquire and release resources according to their needs. However, due to the high cost of context switching, CPU-based serverless frameworks are not well-suited to run thousands of functions. In this work, we propose a solution to mitigate this problem by intelligent offloading of edge computing workloads to the SmartNIC. In particular, we focus on ML workloads at the edge where transient workload spikes can result in SLA violations due to a lack of spare compute resources. There are several challenges and open problems in this area that have not been addressed before. This paper addresses some of these challenges using a generic architecture that can help offload the compute-intensive and network-dependent serverless compute workloads onto the SmartNICs.

2.4 Workload Spikes

We are further motivated for the need for additional compute resources as various datasets and literature illustrate that high CPU utilization lead to over subscription of servers. We also note that high CPU utilization peaks in large scale data centers lead to over subscription of the servers which incur high capital expense (CAPEX) and operational cost (OPEX). For example, Microsoft Azure's dataset from 2019 which contains information of about 2.6M VMs and 1.9B

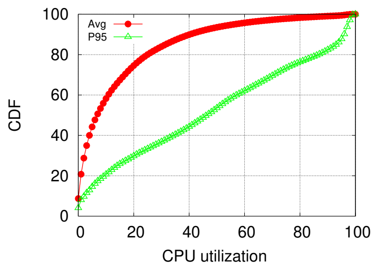


Figure 1: Average and P95 of max CPU utilizations.

utilization data [16] recorded at 5-min time intervals. In Figure 1, we show the Cumulative Distribution Function (CDF) of the average CPU utilization of this dataset and the CDF of 95-th-percentile of maximum CPU utilization (P95 Max). The P95 utilization shows high utilization for almost 62% of VMs. Similarly, in various edge computing scenarios such as CDN edge servers [17], video on-demand [18], enterprise-edge [19], we observe the common trend of requiring high compute resources at the edge to handle high traffic workloads.

While the average CPU utilization is low, P95 CPU utilization is higher than 60% for 62% of the VMs. We note that average utilization is not a good indicator of VMs' compute resource requirements. In Section 4, we show that when the CPU load is high, the average response time latencies of serverless workload gets higher and we see higher SLA violations. To guarantee applications' SLA, cloud and edge providers usually oversubscribe resources which leads to inefficient resource utilization and higher costs. Using SpikeOffload framework, we leverage the extra compute capacity of SmartNICs to manage the P95 of high load spikes to reduce the operational and capital expense costs and reduce SLA violations.

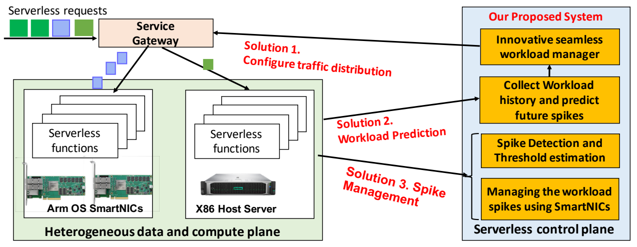
3 SYSTEM OVERVIEW

We begin by providing an overview of SpikeOffload, an intelligent orchestration framework that can be leveraged for any containerized application deployment on data center and/or Edge systems. Figure 2 shows the various components of SpikeOffload framework. The server can host any number of SmartNICs as the number of PCIe buses available. For our PoC system, we use Kubernetes as the container orchestration system that runs on the host and SmartNIC OS. Currently, this specialized architecture works only with SoC-based SmartNIC architecture [6].

The major components of our solution consist of (i) a *workload manager* module that uses the history of the workload in a time window to predict the workload spikes and distributes the request traffic based on the service time and CPU load of each server and SmartNIC, and (ii) the *SpikeOffload orchestrator* module manages the workload spikes based on the load on the servers and SmartNICs. In the following, we describe each module.

3.1 Workload Manager

The current design of Serverless systems and applications is based on the assumption that all computing resource nodes are homogeneous and have the same service time and the same amount of load. This paper shows that this assumption leads to degraded performance



Serverless requests
Service Gateway
Solution 1. Configure traffic distribution
Serverless functions
Serverless functions
Arm OS SmartNICs
X86 Host Server
Heterogeneous data and compute plane
Solution 2. Workload Prediction
Solution 3. Spike Management
Our Proposed System
Innovative seamless workload manager
Collect Workload history and predict future spikes
Spike Detection and Threshold estimation
Managing the workload spikes using SmartNICs
Serverless control plane

Figure 2: System Overview.

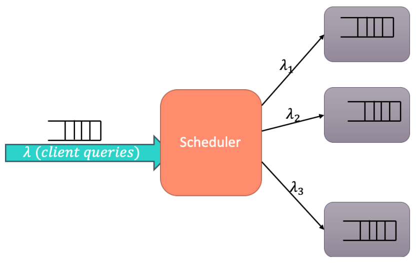


Figure 3: Traffic Distributor.

of workloads running on multiple nodes, especially when one of the compute nodes gets overloaded or takes more time to service the requests. To illustrate the problem, consider two serverless functions A and B that take 2/10 seconds to run on the SmartNIC and 1/5 second to run on the host OS, respectively, but when the host gets overloaded with other workloads, the response times on the host OS changes to 3/8 for functions A and B respectively [1]. In this example, it is better to run the two functions on the host OS when the host OS is not overloaded, and when it gets overloaded, function A can be offloaded to the SmartNIC. This is particularly important in the cases where the workload has data dependency such as with ML workloads and running the workload on an additional server incurs a high cost due to data movement and high latency. To provision for the workload spikes proactively to meet the required Service Level Agreement (SLAs), we predict the future workload demands ahead of time. We propose a support vector regression (SVR) prediction model that predicts the workload bursts to trigger the traffic distribution module and also mitigate the impact of containers' cold start latency [20–23] that can generally lead to a longer response time to application queries otherwise. Our prediction model is based on the past observations of the workload over a window size of W time units. We change the window size dynamically based on the workload variations over time. We increase the training window size if the workload variation over the current window is less than 10% and decreases once the workload variation is more than 20%.

3.2 Traffic Distributor

In our design, the queries first arrive at the API gateway of the scheduler within the SmartNIC OS, further highlighted in Figure 3, where our traffic distributor is located and distributes the traffic

[1]We note that these numbers are subject to change depending on the workload burst and resource congestion on the SmartNICs and host OS servers

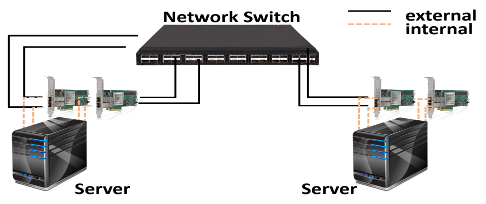


Figure 4: Our evaluation Testbed setup.

according to the service time of each SmartNICs' ARM core or host OS's core within a server. We note that the service time of each function is subject to change depending on the workload spikes. Assuming the requests arrive with the arrival rate of λ and assuming each host OS and SmartNIC have a service rate of μ_i and having an $M/M/1$ queue at each server, the optimal traffic distributor that makes the sojourn time equal for each queue is as follows:

$$\lambda_1 - \mu_1 = \lambda_2 - \mu_2 = ... = \lambda_N - \mu_N \tag{1}$$

In other words, by solving the linear system of equations, the optimal traffic distribution on N servers is as follows:

$$\lambda_i = \mu_i + \frac{\lambda - \sum_{j=1}^{N} \mu_j}{N} \quad i = 1, ..., N \tag{2}$$

where $\lambda = \lambda_1 + ... + \lambda_N$ We continuously monitor the service rates of each node on the cluster and try to avoid running the workload on a cluster node that has very high service time due to workload spikes. The queries are then redirected to the appropriate containerized application pods running either on the Host or SmartNIC OS.

3.3 SpikeOffloadOrchestrator

SpikeOffload consists of a resource monitoring module that exploits the output of the workload manager module. The resource monitoring module periodically monitors each node's CPU and memory utilization and service rates. If the CPU utilization gets higher than a specified threshold Δ, or if the service rate of application X on one of the nodes in the cluster gets higher than the specified SLA, the orchestrator sends feedback to the workload manager to redistribute the workload to dampen the spikes. We use the output of the workload prediction module to predict future spikes ahead of time and perform pro-active spike management. Pro-active spike management has two benefits: (i) first, we can redistribute the traffic based on the predicted future workload, which avoids specific server nodes to get congested, and (ii) second, it mitigates the containers' cold start latency by starting new containers before the actual load arrives. The spike management module updates the service rate, μ_i of each node in the cluster and requests arrival rates in the traffic distributor module, and triggers a new traffic distribution command if the spikes are higher than a specified threshold or the mean service rate of a node in the cluster increases and violate the specified SLA metric.

4 DISCUSSION

In this section, we discuss the benefits and limitations of SpikeOffload based on a quantitative analysis. Avoiding application SLA violation is one of the primary concerns at the edge. We describe how SpikeOffload mitigates SLA violations; we also discuss our testbed setup and performance benefits of SpikeOffload architecture. Then we briefly discuss broader implications and show the challenges we faced to achieve micro-service application offload.

4.1 Edge Computing SLAs

Service Level Agreements are critical when applications are deployed in a Service Oriented Architecture (SOA). SLAs are commonly adopted in cloud computing and, more recently, at the Edge. SLA defines the level of service expected by the consumer based on metrics that the application provider lays out. SLA comprises of the metrics by which the service is measured, such as monitoring the QoS (Quality of Service) metrics [24, 25]. Some of the most common QoS metrics that are part of SLA are response time and throughput. For instance, with ML image classification workload, time taken to classify one image, and the number of images classified per unit of time. In Edge Computing, where there are limited resources when the application receives a burst of queries at an instance, the response time suffers high tail latency. This problem is further strained when the host has additional background workload for other applications or needs part of the edge infrastructure for its Network and Storage needs. This leads to SLA violations and the consumer's poor application Quality of Experience (QoE). In our evaluation, we use the response time metric, to evaluate the penalty with and without additional compute resources such as SmartNICs.

4.2 Performance Benefits

We set up the testbed of SpikeOffload using enterprise-grade high-performance HPE DL380 Gen9 Servers with PCIe 3.0 slots to support two Mellanox Bluefield [26] SmartNICs per server as shown in Figure 4. We deployed a Kubernetes cluster over both server and SmartNIC OS to create a heterogeneous multi-core cluster nodes. To best of our knowledge, this is one of the first efforts to offload a full containerized application onto SmartNICs.

We implemented a prototype the OpenFaaS serverless plaftorm. We evaluated it on three popular serverless workloads, (i) CPU-intensive Fibonacci function, (ii) latency-sensitive key-value store, and (iii) a sentiment analysis function that uses machine learning to perform natural language processing. Since Host and SmartNIC are different compute architectures, We generate multi-architecture images (x86 or ARM-based) for these functions.

Our experiments involve testing the three serverless applications' performance in an environment with transient spikes and heavy background workload with and without offload. We use "***hey***" HTTP(S) load generator [27] to generate incoming queries at scale to the platform and emulate transient spikes using a stress tool [28]. Fig 5. shows the response time distribution for different functions. The SLA threshold is specified by the application user and exposed to the scheduler. We first run the default OpenFaas scheduler on one server, we introduce stress on the host server and increase the average CPU utilization to 80% by running background serverless workload with 200 average queries per second (*Case 1*: 1 server with background workload). The tail latency increases when the host OS has a high load, leading to SLA violations. Adding another server with uniform workload distribution (default Kubernetes scheduler) in the baseline (2 servers, one with background workload and one without background workload) does not solve the problem since half of the

Cluster type	Cost ($)
1 server (case 1)	7000
2 servers (case 2)	14000
1 server and a SmartNIC	8000
1 server and 2 SmartNICs	9000

Table 1: Capital expense costs.

queries are routed to the overloaded host. Next, we run the workload on two servers with proportional workload distribution that is load-aware (*Case 2*: two servers with proportional workload distribution similar to SpikeOffload's workload distributor). In SpikeOffload, we detect the overloaded node in the cluster and avoid routing the workload requests to that node. We run SpikeOffload in two cases, when having one SmartNIC and when we have two SmarttNICs on the same server. Although the SmartNICs have lower computational power than the host OS, when there is a transient spike that overloads the CPU, SpikeOffload leverages SmartNIC's compute capacity to reduce SLA violations.

Figure 6 shows the performance/energy consumption for this scenario. Where we use 1/response time as the performance metric. As shown, running the serverless workload on the SmartNICs provides better performance/energy consumption. The SmartNICs used in our testbed are 3.5x more energy efficient than the host server. As shown, our framework provides more than 2x performance/energy saving. We also note that there is obvious cost efficiency when using SmartNICs given its lower cost as compared to servers as shown in Table 1.

4.3 Broader Implications

There has been a tremendous rise in the adoption of Function-as-as-Service (FaaS) or Serverless, potentially due to their economically attractive services with reduced operational costs compared to Infrastructure-as-a-Service (IaaS). Enterprises like ClearBlade, EdgeConneX, and Edge Intelligence focus on running FaaS workloads at the edge instead of processing data at a data warehouse. We believe a platform like SpikeOffload that leverages DPUs (e.g. SmartNICs) for offloading transient spikes and smaller workloads could significantly improve the scalability and efficiency of edge computing resources at a fraction of the cost of adding bare-metal servers, as is traditionally done.

Challenges in DPU Offload

In this section, we enumerate some of the challenges we faced and lessons learned while developing SpikeOffload.

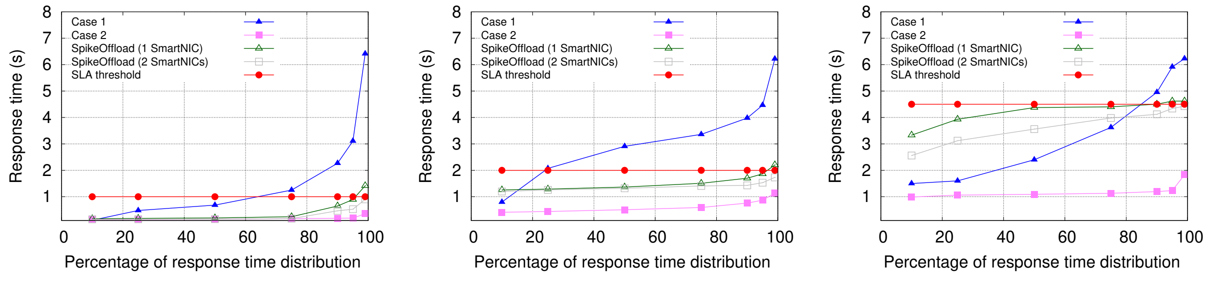
Hardware-based: Manufacturers have yet to make SmartNICs universally compatible with all available bare-metal options. Additionally, to scale a platform like SpikeOffload, it is imperative to have a vendor-agnostic solution. For instance, SoC-based SmartNICs could offer a universal protocol to set up Linux environments effortlessly; they are currently tightly coupled into their vendor-designed architectures, which involves procedural effort for initial setup. However, we note that this is relatively easier to achieve than setting up bare-metal servers that require infrastructure-based protocols such as iLo[29] for automated setup.

SoC-based SmartNICs, which are generally "off-path" in architecture design, contain a "nic-switch" [9] that determines the packet's path once it enters the physical interface. This "nic-switch" has a rigid design based on the vendor's architecture, and therefore, latency performance is skewed for embedded or separated modes. Moreover, most SmartNIC vendors only enable embedded mode, which further challenges the deployment of SpikeOffload framework. Enabling flexibility of the "nic-switch" can potentially lower latency and improve the performance of SpikeOffload.

Software-based: SoC-based SmartNICs typically embed an ARM64 architecture, while bare-metal solutions usually are built on an x86 architecture. This heterogeneity in system architecture requires us to develop and make available serverless applications (container images) for each system type (ARM65 and x86) – leading to higher temporal overheads. Furthermore, SpikeOffload framework is developed for SoC-based SmartNICs [9, 11], given the need for a Linux-based operating system to run container orchestration (that executes the workloads). Further research is required to develop SpikeOffload-like solutions for ASIC- and FPGA-based SmartNICs. SoC-based SmartNICs (e.g., Mellanox) have two modes of operation: *Embedded*, and *Separated* modes. The interfaces are mapped to the host OS network stack in embedded mode and the kernel routes packets from the host. The host OS and the SmartNIC have separate, independent network stacks to process packets in the separated mode. While we observe slightly better tail-latencies from packet processing in embedded mode, the offset from separate mode is negligible. For SpikeOffload, we adopt the separated mode due to its programmable flexibility and the ability to run containers directly on the SmartNIC's ARM64 OS. We observe that the recent versions of SmartNICs can support containers with many network restrictions. We hope that SpikeOffload can work with both modes with more improvements from the vendor in the underlying hardware or by providing the compatible container network interface (CNI). Previous work [9] has demonstrated the efficiency gain in offloading networking functions onto the SmartNICs. However, SmartNICs have limited computation capacity when compared to server OS. For example, SmartNIC in our testbed has 16 processing threads with 312.50 flops, and the host server has 40 processing threads with 4604.74 available flops. So, modules (in kernel space) that prioritize networking functions over application workloads are necessary to ensure proper functionalities of network processes. Moreover, careful monitoring is needed to ensure the area under the curve (AUC) for workload spikes does not exceed the compute capacity of SmartNICs (serverless functions typically have smaller footprints). Lastly, to minimize cold-start latency of the initial offload (potentially have higher tail-latencies), seed pods need to be warmed up on the offload devices at all times. This approach can incur a small computation overhead.

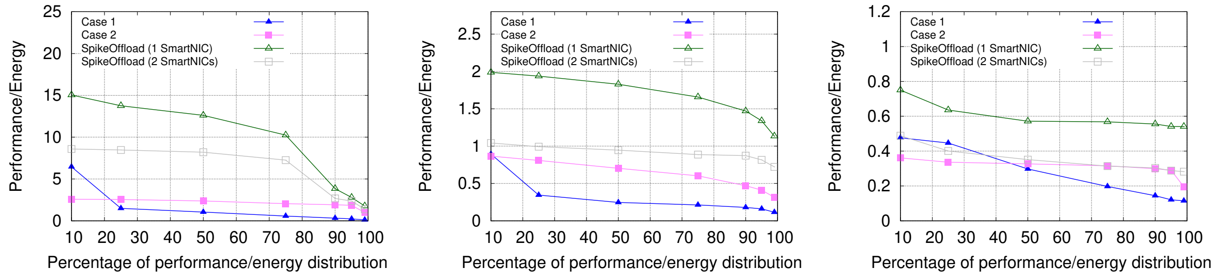
5 RELATED WORK

Recent research on SmartNICs focuses on leveraging the compute power to offload various application workloads [8–10, 30]. Most of these approaches focus on moving small functions onto the SmartNIC OS. At the same time, SpikeOffload is a framework that can offload complete workload containers onto the SmartNIC OS, leveraging the separated-host mode (Sec 2.1) of SoC-based SmartNICs. Most other research on SmartNIC hardware offloading is limited to network functions such as load-balancing, firewall, etc. [31]. In SpikeOffload, we differentiate by offloading application workloads



(a) Fibonacci. (b) Key-value store. (c) Sentiment analysis.

Figure 5: Response time distribution of different functions.



(a) Fibonacci. (b) Key-value store. (c) Sentiment analysis.

Figure 6: Performance/Energy consumption distribution of different functions.

as containers, not just functions. We showcase the benefit and also explore the challenges.

Workload spike management is a well-explored problem space in the cloud and networks [2, 32] , where user query surge can lead to downtime and poor QoE. At the edge, it can be pretty challenging to deal with traffic spikes, and the most common business solution is over-provisioning computing resources [33, 34]. We differentiate by utilizing SmartNICs to address transient spikes at the edge, thereby enabling transient elasticity of resources. SmartNICs are limited in computing capacity, so it is critical to offload essential workloads. Therefore, SpikeOffload's approach to offload only during traffic spikes validates the strategy given the CAPEX and OPEX savings without much application performance degradation. In that aspect, Serverless applications are gaining popularity to be deployed at the edge for AI, security, and storage workloads [35, 36].

Certain body of research have explored offloading functions of applications onto SmartNICs processing units[8–10]. For instance, Lambda-NIC [8] demonstrates offloading data plane programming functionality of serverless applications to ASIC SmartNICs. While iPipe [9] offloads applications designed in actor-programming model.

In SpikeOffload, we adopt a novel approach of offloading the entire containerized serverless application (small function containers) onto the SoC-based SmartNIC OS by establishing SmartNIC OS as nodes in the cluster network. While we move the whole container to the Smartnic, [8] rely on P4 programmability to offload a small part of applications to the SmartNic. In [8] the host and the Smartnic are one single node in the Kubernetes cluster, and changing the application requires code modification to offload to the SmartNic, while in our framework, the SmartNic is one of the nodes in the cluster and can leverage the Kubernetes orchestration system for scheduling, auto-scaling, etc.

6 CONCLUSION

This paper proposes a new platform that leverages the DPUs' and SmartNICs' computational capacity to offload and accelerate serverless workload in the presence of transient traffic spikes at a lower cost. Our solution is three-fold. First, we propose a novel system architecture leveraging container orchestration systems to distribute the workloads between Hosts and SmartNICs based on the demand for transient elasticity of resources. The next challenge we solve is to manage the workload spikes by exploiting the unused computational capacity of the SmartNICs to avoid SLA violations. Finally, we propose a novel workload prediction approach that predicts the transient spikes and starts the containers before the actual load arrives in the system to mitigate the containers' cold start latency. Accounting for transient elasticity using SmartNICs has the added benefit of provisioning a hybrid cloud and edge deployment, with the flexibility to scale edge deployments when required. This could lead to faster turnaround times for system administrators executing decisions to allocate compute cycles. While this paper focuses on transient elasticity for workload spikes, SpikeOffload architecture can be leveraged for building a generalized system for federated edge infrastructure with heterogeneous DPUs like GPU and SmartNICs, etc.

REFERENCES

[1] Z. Zhong, N. Hua, M. Tornatore, J. Li, Y. Li, X. Zheng, and B. Mukherjee. Provisioning Short-term Traffic Fluctuations in Elastic Optical Networks. *IEEE/ACM Transactions on Networking*, 2019.

[2] Z. Liu, X. Wang, W. Pan, B. Yang, X. Hu, and J. Li. Towards Efficient Load Distribution in Big Data Cloud. In *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015.

[3] A. Ali-Eldin, O. Seleznjev, S. Sjöstedt-de Luna, J. Tordsson, and E. Elmroth. Measuring cloud workload burstiness. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 566–572, 2014.

[4] Azure Functions Serverless Architecture — Microsoft Azure. https://azure.microsoft.com/en-us/products/functions/. [Online; accessed 2019].

[5] GCP uses Intel Network Accelerators. link. [Online; accessed 2022].

[6] Mellanox Bluefield. link. [Online; accessed 2021].

[7] Broadcom Stingray. link. [Online; accessed 2021].

[8] M. Choi, S.and Shahbaz, B. Prabhakar, and M. Rosenblum. λ-NIC: Interactive Serverless Compute on Programmable SmartNICs. *arXiv preprint arXiv:1909.11958*, 2019.

[9] M. Liu, T. Cui, H. Schuh, A. Krishnamurthy, S. Peter, and K. Gupta. iPipe: A Framework for Building Distributed Applications on Multicore SoC SmartNICs. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2019.

[10] M. Liu et al. E3: Energy-efficient Microservices on SmartNIC-accelerated Servers. In *USENIX ATC*.

[11] D. Firestone et al. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018.

[12] Netronome Agilio CX. link. [Online; accessed 2021].

[13] SmartNICs, the Next Wave in Server Acceleration. link. [Online; accessed 2020].

[14] Cloud Functions - Serverless Environment to Build and Connect Cloud Services — Google Cloud Platform. https://cloud.google.com/functions/. [Online; accessed 2019].

[15] Y. Cui. I'm afraid you're thinking about AWS Lambda cold starts all wrong. link. [Online; accessed 2019].

[16] Microsoft Azure's Public Dataset. https://github.com/Azure/AzurePublicDataset/, 2021.

[17] Traffic spike mitigation from akamai. https://developer.akamai.com/article/traffic-spike-mitigation. [Online; accessed 2021].

[18] P. Gill, M. Arlitt, Zo. Li, and A. Mahanti. Youtube Traffic Characterization: A View from the Edge. IMC '07. Association for Computing Machinery, 2007.

[19] S. Wang. Edge computing: Applications state-of-the-art and challenges. *Advances in Networks*, 2019.

[20] L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift. Peeking behind the Curtains of Serverless Platforms. In *USENIX Annual Technical Conference (ATC)*, 2018.

[21] I. E. Akkus, R. Chen, I. Rimac, M. Stein, K. Satzke, A. Beck, P. Aditya, and V. Hilt. SAND: Towards High-Performance Serverless Computing. In *2018 USENIX Annual Technical Conference*, pages 923–935, 2018.

[22] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Serverless Computation with OpenLambda. In *HotCloud*, 2016.

[23] W. Lloyd, S. Ramesh, S. Chinthalapati, L. Ly, and S. Pallickara. Serverless computing: An investigation of factors influencing microservice performance. In *International Conference on Cloud Engineering (IC2E)*. IEEE, 2018.

[24] K. Katsalis, T. G. Papaioannou, N. Nikaein, and L. Tassiulas. SLA-driven VM Scheduling in Mobile Edge Computing. In *IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016.

[25] P. Patel, A. H. Ranabahu, and A. P. Sheth. Service Level Agreement in Cloud Computing. 2009.

[26] Mellanox Bluefield. link, 2020.

[27] Hey, HTTP(S) load generator, ApacheBench (ab) replacement, written in Go. https://github.com/rakyll/hey. [Online; accessed 2019].

[28] Stress - Tool to Impose Load on and Stress Test Systems. link. [Online; accessed 2021].

[29] Alexandre Gazet, Fabien Périgaud, and Joffrey Czarny. HPE iLO 5 security: Go home cryptoprocessor, you're drunk!

[30] Y. Qiu, Q. Kang, M. Liu, and A. Chen. Clara: Performance Clarity for SmartNIC Offloading. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, 2020.

[31] J. Kicinski and N. Viljoen. eBPF Hardware Offload to SmartNICs: cls bpf and XDP. *Proceedings of netdev*, 2016.

[32] A. Mehta, J. Dürango, J. Tordsson, and E. Elmroth. Online Spike Detection in Cloud Workloads. In *IEEE international conference on cloud engineering*, 2015.

[33] V. Nikolov, S. Kächele, F. J. Hauck, and D. Rautenbach. Cloudfarm: An Elastic Cloud Platform with Flexible and Adaptive Resource Management. In *IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014.

[34] A. Gandhi, T. Zhu, M. Harchol-Balter, and M. A. Kozuch. SOFTScale: Stealing Opportunistically for Transient Scaling. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2012.

[35] L. Baresi and D. Filgueira Mendonça. Towards a Serverless Platform for Edge Computing. In *2019 IEEE International Conference on Fog Computing (ICFC)*, 2019.

[36] Muthusamy V Rashed A Dustdar S Rausch T, Hummer W. Towards a serverless platform for edge AI. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2019.