SmartNICs in the Cloud: The Why, What and How of In-network Processing for Data-Intensive Applications

Faeze Faghih
faeze.faghih@tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Tobias Ziegler
tobias.ziegler@tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Zsolt István
zsolt.istvan@tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Carsten Binnig
carsten.binnig@tu-darmstadt.de
Technical University of Darmstadt & DFKI
Darmstadt, Germany

ABSTRACT

In modern datacenters and clouds, Resource Disaggregation has been adopted as a way of offering scalability and efficient resource utilization for large-scale applications. Provisioning CPU, memory, and storage resources independently for distributed data-intensive applications is a great enabler but it also brings challenges, especially in the form of networking and processing overhead. To reduce this overhead and to make disaggregation-related tasks significantly more efficient, cloud providers are offloading these to the network, i.e., to Smart Network Interface Cards (SmartNICs) and Smart Switches. Beyond this specific use-case, the presence of such programmable hardware in commercial clouds creates future opportunities for offloading application-level operations, e.g., parts of SQL queries or ML pipelines. To map out this exciting space, in this tutorial we take a detailed look at SmartNICs, explaining how they work, giving examples of what they are good for, and highlighting how they can best be utilized to make future data-intensive and distributed systems more efficient.

CCS CONCEPTS

• **Hardware → Networking hardware**; • **Networks → Programmable networks**; • **Information systems → Database query processing**.

KEYWORDS

Programmable NICs; FPGAs; Database Management Systems

ACM Reference Format:
Faeze Faghih, Tobias Ziegler, Zsolt István, and Carsten Binnig. 2024. SmartNICs in the Cloud: The Why, What and How of In-network Processing for Data-Intensive Applications. In *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24), June 9–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626246.3654690

1 MOTIVATION AND PURPOSE

Modern data-intensive applications, such as analytical database management systems or machine learning pipelines, face several scalability challenges. Single node compute performance and memory capacity is not enough for most use-cases and, as a result, these applications are commonly run as distributed systems, on multiple nodes in the Cloud. In order to make more efficient use of compute, storage, and memory resources, and to allow right-sizing them for each application, new mechanisms have been introduced in modern clouds and datacenters to disaggregate these resources.

Disaggregated architectures expose different types of resources over the network and, even though the network connection speeds are steadily increasing (already reaching 100Gbps speeds in clouds with 200Gbps and 400Gbps on the horizon), disaggregation is still facing challenges on the CPU-side that limit scalability. In a CPU-centric design, a significant portion of CPU resources has to be spent on operations related to the access and management of the disaggregated infrastructure, such as, data serialization, network virtualization, etc. With increasing network speeds, the stagnation of modern CPU performance [10] leads to a clear bottleneck.

Related work coined the term "datacenter tax" [14, 18] to describe the CPU cycles spent not on user applications but to enable infrastructure-related features, such as resource disaggregation. One way to reduce the datacenter tax is to move parts of the infrastructure code away from the CPU and to "offload" it to the network, where it can be executed more efficiently. Even application code can benefit from offloading closer to the network: for instance, SQL filtering done closer to the data source will result in overall better network bandwidth utilization.

One way to move processing closer to the network is by using Smart Network Interface cards (SmartNICs)[1], which are NICs with a specialized hardware component added to them. The specialized component is designed to be very efficient on a small set of recurring tasks and enables efficient close-to-network computation that frees up general purpose CPU cycles. SmartNICs are becoming common in the cloud and in datacenters, with success stories at the cloud vendors, e.g., in handling network virtualization tasks

[1]Smart Switches are another option for processing in the network, but we decided not to cover them in this tutorial. By looking at SmartNICs only, we can cover in detail their inner working, opportunities for databases, and needs for change in the future. Actually, a large subset of possible use-cases for Smart Switches can also be implemented using SmartNICs at the end host.

more efficiently than CPUs in the Azure Cloud [9], or by offloading SQL operations closer to the distributed storage in the (now discontinues) AWS AQUA processors [17]. They are not yet part of the database community, but there are clear opportunities for using SmartNICs to make Cloud DBMSs more efficient.

This tutorial has a dual purpose. On the one hand, it explains the motivation behind SmartNICs and their design options, highlighting how they emerged as an answer to the challenges that modern clouds have faced in the last decade. On the other hand, it shows how SmartNICs can be successfully used in DBMSs, through examples from our own research and from related work. For instance, in our previous work [21] we saw that CPU-based SmartNICs which use weak cores cannot provide the expected benefits while other platforms, incorporating specialized hardware, could significantly speed up database operations. In this tutorial, we provide the necessary background for understanding what platforms are available, which ones are suitable for disaggregated DBMSs in the cloud, and how we can design better DBMS-centric SmartNICs.

2 TUTORIAL CONTENTS

The tutorial is structured in three parts. Part 1 covers the main motivation behind processing closer to the network and the explanation of why SmartNICs are already being deployed in the cloud. Afterward, Part 2, covers the design space of SmartNIC hardware, relating their possible use-cases to different architectural alternatives, clarifying for the audience what they can and cannot do in terms of offloading. Part 3 covers how SmartNICs can be used in database scenarios, with an overview of a handful of projects carried out in our group and in related work. In the third part we also provide a small toolbox for database researchers to determine whether, for a specific workload, offloading to the NIC will be beneficial or not. The main contribution of the tutorial is to bring these three parts together, covering the motivation, the design space, and the use-case considerations of SmartNICs in an in-depth fashion.

2.1 Part 1: The Problem SmartNICs are Solving

Modern Clouds and Disaggregation. The computing landscape has shifted in the last decades, with more and more companies running their data-intensive operations in clouds or enterprise data centers that are designed and operated like a public cloud [2]. This transition has made many Big Data applications, such as Cloud DBMSs or distributed machine learning pipelines, feasible. But even though applications can theoretically scale to a very large number of nodes in the cloud, challenges remain when it comes to efficiently utilizing all resources in a cloud environment.

It has been shown in multiple studies [6, 15] that course-grained provisioning of cloud resources to applications leads to underutilized components. As a result, cloud architectures have been moving towards disaggregation: processors, memory, and storage resources can be provisioned independently of each other, and modern application design has also followed suit in being more distributed [1, 19].

Disaggregation is achieved, to a large extent, by making the different resources of traditional cloud nodes accessible over the network. Even though disaggregation brings many benefits, it requires lots of data movement across nodes and has its scalability challenges. High bandwidth networks (with 100Gbps or faster speeds) are an important enabler of disaggregation but they are not the only component to consider: disaggregation requires management, orchestration, and data movement operations in software. These operations all use a significant fraction of CPU cycles on the cloud nodes [14]. In itself, this would not be a problem, but these CPU cycles cannot be monetized anymore as VMs or as services of the cloud provider. In a research paper by Google, the CPU overhead of disaggregation-related operations was coined as the "datacenter tax" [14, 18]. In light of stagnating CPU performance [10] and rising network speeds, the datacenter tax remains the main hurdle for further scalability in the cloud.

SmartNICs to reduce the Datacenter Tax. The challenges of providing efficient disaggregated architectures can be addressed in modern clouds by relying on specialization as a way of reducing the datacenter tax and freeing up CPU cycles. Smart Network Interface Cards are already available from hardware vendors, such as NVIDIA [3] or AMD [5], and are also being actively deployed in the cloud.

One SmartNIC success story is the Azure Accelerated Networking project [9], that uses SmartNICs to solve a network virtualization problem. At Microsoft, as at other cloud providers, VMs belonging to different tenants run on the same physical infrastructure and they must be isolated. This isolation includes their network traffic, both for performance and for security reasons. Software Defined Networking (SDN) is a common way to achieve this isolation, which relies on network packet re-writing rules. In Microsoft Azure, these rules were applied at the CPU, which lead to a portion of the CPU being dedicated to this task, instead of being monetized as user VMs. The solution that was adopted consists of a specialized hardware-based NIC that combines regular network processing with programmable SDN rule evaluation to achieve per-flow rule execution in the network card. This design reduces CPU overhead to almost zero (thanks to the more efficient specialized hardware-based execution) and achieves overall lower latency and higher bandwidth than the CPU-only version. This project also shows that this approach is economically feasible: Microsoft has been deploying SmartNICs on all new servers in the Azure cloud since 2015. In addition to SDN rule offloading, these SmartNICs are used for other projects as well, where they help in making application-level operations more efficient, such as Machine Learning Inference [4].

2.2 Part 2: The Internals of SmartNICs

What's inside a NIC? Network Interface Cards (NICs) are something that most of us "take for granted" but they play a crucial role in enabling communication over the network at high bandwidths and low latency. In order to understand the design space of SmartNICs, the tutorial will start with a high level overview of what tasks "simple" NICs perform (including handling physical access to the network, implementing networking protocols, and virtualization capabilities) and how they are built at the hardware level. As Figure 1 illustrates, the various steps performed in the NIC are conceptually connected with operations at different layers of the hardware/software architecture. The fact that data packets move up/down different levels of abstraction, from bits on the wire, through OS level concepts (like data streams), to application data,

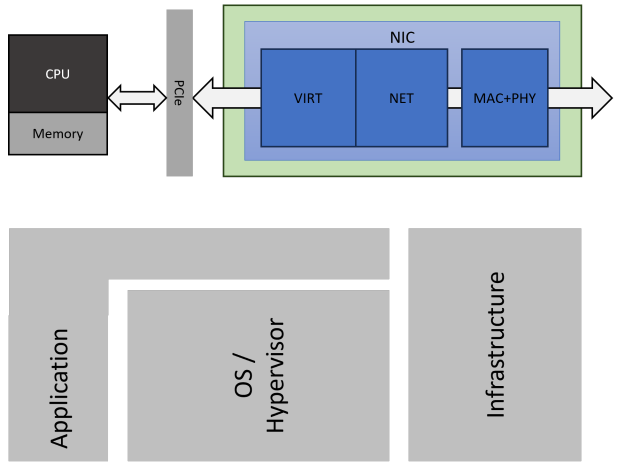


Figure 1: Network Interface Cards, even without the "smarts", are already in charge of a number of important operations, related to different conceptual layers. As we later explain, the location where a Smart Component is inserted into the NIC architecture defines at what conceptual layer it can provide efficient offloading.

will be important for discussing the benefits and drawbacks of including programmable elements in various places in the NIC.

There's more than one way to build a SmartNIC. When adding a Smart Component to a NIC, there are two dimensions to consider: (a) the type of compute element that executes the "smarts" and (b) the location of this element in the NIC architecture. Both dimensions are important and, together, they determine for which use-cases a SmartNIC will be useful.

For the type of compute element added to the SmartNIC, usually one of the following three options is used (in increasing order of flexibility and decreasing order of energy efficiency):

(1) Application Specific Integrated Circuits (ASICs) – This choice of compute element will result in the highest performance and the highest energy efficiency but has the least flexibility. This is because ASICs can only offer a fixed set of functions, which are decided at design time and cannot be changed later. Examples from real-world devices include functions that are often used in the same way by many applications, e.g., encryption/decryption with commonly used ciphers.
(2) Reprogrammable Hardware, such as Field Programmable Gate Arrays (FPGAs) – these represent a middle ground between performance and flexibility. FPGAs are hardware chips that are composed of a collection of small look-up tables (LUTs), on-chip memory (BRAM) and specialized digital signal processing units (DSPs), which can be configured and interconnected to implement any hardware circuit. In comparison to traditional processors, FPGAs allow for fine-grained dataflow parallelism due to the fact that all "code" executes in parallel inside the device. The energy footprint of FPGAs is at least order of magnitude lower than that of server-grade CPUs; and even though it is higher than that of ASICs, FPGAs can be reprogrammed freely, whereas ASICs have fixed functionality.
(3) Low Power CPU cores (such as ARM cores) – this option brings the most flexibility but it is also the one with the lowest nominal performance. Since NICs have to remain low power devices in the computer's architecture, the processors running on them cannot be clocked at very high frequency, nor can they have many cores. As a result, SmartNICs with wimpy CPU cores on them are useful only when performing light-weight computation close to the network is already a benefit – if more heavy-weight computation is needed, the CPU-based compute element on the SmartNIC will become the bottleneck.

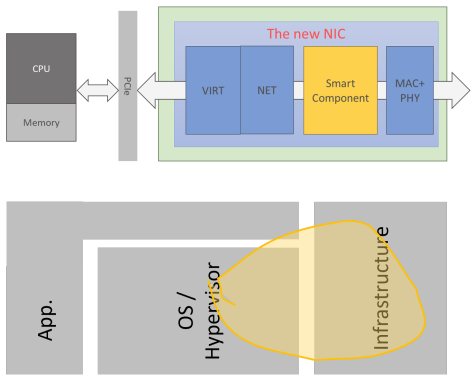
In terms of the location of the Smart Component in the NIC there are two main options, also depicted in Figure 2. The choice of where the Smart Component is deployed will dictate what kind of offloading a specific design can handle most efficiently:

- When deployed "closer to the network wire", between the network and what we consider the traditional NIC logic, the Smart Component can best handle operations that perform network protocol processing, packet transformations, or provide virtualization. The Smart Component is inserted as a streaming processor between the layers of the NIC and has to be designed to operate at the network line-rate, independent of the data that it is processing – as such, in this architecture, usually either an ASIC or an FPGA is used (see, for instance, Azure Accelerated Networking [9] and also research platforms [7, 22]).
- When deployed "closer to the CPU", between the Host CPU and the core of the NIC, the operation (and management) of the Smart Component is decoupled from the operations of the NIC. This architecture is typically achieved by adding a PCIe switch inside the NIC, to which the Smart Component is connected. This results in a lot of flexibility for application-level offloading because the host CPU can communicate with the NIC or the offloading logic separately, as well as use the Smart Component as a streaming processor on all outgoing/incoming packets. The drawback of this design is that the PCIe bus can become a performance bottleneck and bandwidth/latency guarantees cannot always be provided. This approach is adopted by most commercial SmartNICs, e.g., NVIDIA Bluefield [21] (but also SmartSSDs [16]), due to its simplicity in terms of hardware design.

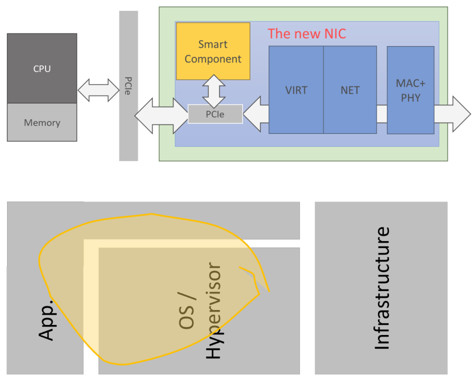
The most important take-away in this part of the tutorial is that the location of the smart component in a NIC will determine at what level of abstraction it can provide offloading (whether at the network packet level, application session level, etc.) and what relative performance it can achieve when compared to a regular CPU that it should offload computation from.

2.3 Part 3: Using SmartNICs in Databases

How can databases benefit? Based on the lessons from the previous parts of the tutorial, we will discuss what operators and what tasks can benefit from SmartNICs in the context of DBMSs on disaggregated cloud infrastructure. We will show what is the state of the art in terms of using SmartNICs in such scenarios and what are



a) Offloading takes place "closer to the network", between the network wire and the traditional NIC tasks. This architecture works best for offloading lower-level operations, since application-level concepts (e.g., encryption keys) are cumbersome to expose to the Smart Component.



b) Offloading takes place "closer to the CPU", between the traditional NIC tasks and the host CPU. This architecture is a good match for offloading high-level operations, as applications can access and manage the Smart Component directly and can expose high-level concepts to it.

Figure 2: SmartNICs can incorporate smartness at various places in their architecture. In the tutorial, we focus on two common designs of commercially available devices and highlight what kind of offloading they are best suited for.

future challenges (at a high level) that will have to be overcome. As concrete examples of offloading from different conceptual levels, we will present the following:

First, using our previous work on implementing processing near the network in the distributed storage layer [13] as an example, we will discuss which parts of analytical SQL queries can be offloaded to SmartNICs, using what interface, and how does the Smart Component design for the offloaded operations look like (as an example of application-level offloading). We will focus on Selection filters and Group By aggregation, because they result in the most reduction of data sizes and can be carried out in parallel at each partition of the data in distributed storage. We will use this example to provide an intuition to the audience on how algorithms, that are compute-bound on traditional CPU architectures, can be turned into bandwidth-bound ones on FPGAs.

As a second example, we will present our findings on the topic of offloading data access operations to a SmartNIC equipped with ARM cores [21], discuss its limitations, and show the interplay with faster networking technologies, i.e., RDMA. We will discuss what opportunities and challenges are there when traversing B-Tree indexes using RDMA and with offloading on the SmartNIC.

In addition to the two application-level offloading examples, we will also discuss a third opportunity we identified for tailoring the infrastructure-level operations for database needs. Namely, we sketch how RPC calls used in a Cloud DBMS could be made faster thanks to offloading to a SmartNIC – but we will also highlight the shortcomings of some of the SmartNICs on the market today in achieving this goal.

Deciding when to offload and what to offload. The tutorial will provide a "rule of thumb" to help decide which operations to offload from a query workload, and how to determine the target processing rate of the SmartNIC. This is important because working with specialized hardware requires extra effort and unless the offloading results in significant performance/efficiency gains, it might not be worth it. To this end, we will present our early design for a framework for modeling the behavior of streaming data analytics platforms [8], with the goal of determining which query steps should be offloaded to SmartNICs and what speed this offload has to achieve. In this framework, the query processing is represented as a network of queues, where for each communication link and processing step, we determine its average bandwidth and its data reduction factor (e.g., selectivity of a filter predicate), and then use these to pinpoint bottlenecks. Even though our model requires only a few measurements to be built, it can help design offloading in a wide range of workloads and targeting different SmartNIC types.

Benefits of SmartNICs beyond query processing. If time permits, we will use the last part of the tutorial for an example of how SmartNICs could be used not only to make existing query-processing tasks more efficient, but to allow us to add new features to the database or to the infrastructure without slowing them down. Our previous work on Software Defined Data Protection [12] is an example of using in-network processing to increase the privacy/security guarantees of the storage layer of a distributed database, without slowing query processing down.

3 ORGANIZATION AND FURTHER DETAILS

3.1 Duration and Style

The length of the tutorial is three hours. It is a mix between a primer and a survey: It is like a primer because we expect that the topic of NIC internals and SmartNIC hardware design will be relatively new for most audience members. The discussion of success stories in the DBMS setting is more like a survey and a compass to understand what works well in which Cloud DBMS scenario. Overall, our goal is to help audience members understand when SmartNICs can be useful for DBMSs, while also providing explanations at the hardware level on why this is the case.

3.2 Target Audience and Prerequisites

Our target audience is a mix of researchers and practitioners, with a good basic understanding of how data-intensive systems work at the software level. There are no assumed prerequisites in cloud architecture or in computer architecture. At the same time, given that we will provide a holistic view of using SmartNICs in Cloud DBMS, we believe that the tutorial will be useful even for those audience members who are already knowledgeable in using modern network technologies.

3.3 Presenter Bios

Faeze Faghih is a doctoral student in the Systems Group at TU Darmstadt. Her expertise is in programmable and heterogeneous hardware, and she currently works on performance modeling of Smart NIC and Smart Storage-equipped data processing systems [8].

Tobias Ziegler is a postdoctoral researcher in the Systems Group at TU Darmstadt. He specializes in designing and developing scalable, high-performance data management systems. His primary research interest lies in utilizing modern network technologies, including RDMA (Remote Direct Memory Access) and programmable Network Interface Cards (NICs), to improve the efficiency and performance of distributed databases. His recent work on RDMA has been awarded a SIGMOD Best Paper award in 2021.

Zsolt István is a Full Professor in the Systems Group at TU Darmstadt. In his research, he focuses on distributed and networked systems topics. Earlier, he was an Associate Professor at the IT University of Copenhagen, Denmark, and an Assistant Research Professor at the IMDEA Software Institute in Madrid, Spain. Zsolt received his PhD degree from the Systems Group at ETH Zurich, Switzerland. He has contributed to several projects using FPGAs as SmartNICs [20] and to disaggregate cloud resources [13]. He has co-authored a book on FPGA-accelerated Databases [11].

Carsten Binnig is a Full Professor in the Systems Group at TU Darmstadt and a Visiting Researcher at the Google Systems Research Group. Carsten received his Ph.D. at the University of Heidelberg in 2008. Afterwards, he spent time as a postdoctoral researcher in the Systems Group at ETH Zurich and at SAP working on in-memory databases. Currently, his research focus is on the design of scalable data systems on modern hardware as well as machine learning for scalable data systems. His work has been awarded a Google Faculty Award, as well as multiple best paper and best demo awards.

ACKNOWLEDGMENTS

Our work presented in this tutorial has been partially funded through a Research Grant by Huawei Research Munich, by the German Research Foundation (DFG) under the grants BI2011/1 & BI2011/2 (DFG priority program 2037), and the DFG Collaborative Research Center 1053 (MAKI).

REFERENCES

[1] Audibert, A., Chen, Y., Graur, D., Klimovic, A., Simsa, J., and Thekkath, C. A. A case for disaggregation of ml data processing. *arXiv preprint arXiv:2210.14826* (2022).

[2] Barroso, L. A., Hölzle, U., and Ranganathan, P. *The datacenter as a computer: Designing warehouse-scale machines.* Springer Nature, 2019.

[3] Burstein, I. Nvidia data center processing unit (dpu) architecture. In *2021 IEEE Hot Chips 33 Symposium (HCS)* (2021), IEEE, pp. 1–20.

[4] Chung, E., Fowers, J., Ovtcharov, K., Papamichael, M., Caulfield, A., Massengill, T., Liu, M., Lo, D., Alkalay, S., Haselman, M., et al. Serving dnns in real time at datacenter scale with project brainwave. *iEEE Micro 38*, 2 (2018), 8–20.

[5] Dastidar, J., Riddoch, D., Moore, J., Pope, S., and Wesselkamper, J. The amd 400-g adaptive smartnic system on chip: A technology preview. *IEEE Micro 43*, 03 (2023), 40–49.

[6] Delimitrou, C., and Kozyrakis, C. Quasar: Resource-efficient and qos-aware cluster management. *ACM SIGPLAN Notices 49*, 4 (2014), 127–144.

[7] Eran, H., Zeno, L., Tork, M., Malka, G., and Silberstein, M. {NICA}: An infrastructure for inline acceleration of network applications. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)* (2019), pp. 345–362.

[8] Faghih, F., István, Z., and Dinu, F. The next 700 heterogeneous olap systems: A framework to answer what-if design questions. Poster Session of ACM EuroSys'23 Conference.

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

[10] Hennessy, J. L., and Patterson, D. A. A new golden age for computer architecture. *Communications of the ACM 62*, 2 (2019), 48–60.

[11] István, Z., Kara, K., Sidler, D., et al. Fpga-accelerated analytics: From single nodes to clusters. *Foundations and Trends® in Databases 9*, 2 (2020), 101–208.

[12] István, Z., Ponnapalli, S., and Chidambaram, V. Software-defined data protection: Low overhead policy compliance at the storage layer is within reach! *Proceedings of the VLDB Endowment 14*, 7 (2021), 1167–1174.

[13] István, Z., Sidler, D., and Alonso, G. Caribou: Intelligent distributed storage. *Proceedings of the VLDB Endowment 10*, 11 (2017), 1202–1213.

[14] Kanev, S., Darago, J. P., Hazelwood, K., Ranganathan, P., Moseley, T., Wei, G.-Y., and Brooks, D. Profiling a warehouse-scale computer. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture* (2015), pp. 158–169.

[15] Klimovic, A., Kozyrakis, C., Thereska, E., John, B., and Kumar, S. Flash storage disaggregation. In *Proceedings of the Eleventh European Conference on Computer Systems* (2016), pp. 1–15.

[16] Lee, J. H., Zhang, H., Lagrange, V., Krishnamoorthy, P., Zhao, X., and Ki, Y. S. Smartssd: Fpga accelerated near-storage data analytics on ssd. *IEEE Computer architecture letters 19*, 2 (2020), 110–113.

[17] Pandis, I. The evolution of amazon redshift. *Proceedings of the VLDB Endowment 14*, 12 (2021), 3162–3174.

[18] Seemakhupt, K., Stephens, B. E., Khan, S., Liu, S., Wassel, H., Yeganeh, S. H., Snoeren, A. C., Krishnamurthy, A., Culler, D. E., and Levy, H. M. A cloud-scale characterization of remote procedure calls. In *Proceedings of the 29th Symposium on Operating Systems Principles* (2023), pp. 498–514.

[19] Shahrad, M., Fonseca, R., Goiri, I., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Tresness, C., Russinovich, M., and Bianchini, R. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX annual technical conference (USENIX ATC 20)* (2020), pp. 205–218.

[20] Sidler, D., István, Z., and Alonso, G. Low-latency tcp/ip stack for data center applications. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)* (2016), IEEE, pp. 1–4.

[21] Thostrup, L., Failing, D., Ziegler, T., and Binnig, C. A dbms-centric evaluation of bluefield dpus on fast networks. In *13th International Workshop on Accelerating Analytics and Data Management Systems Using Modern Processor and Storage Architectures* (2022).

[22] Zilberman, N., Audzevich, Y., Covington, G. A., and Moore, A. W. Netfpga sume: Toward 100 gbps as research commodity. *IEEE micro 34*, 5 (2014), 32–41.