”ZeroCopy-DSMS: A Zero-copy, CPU-efficient Serverless Architecture using Distributed Shared Memory and SmartNICs”

Tech Con 2025 Abstract 729

Hybrid Cloud;

ZeroCopy-DSMS: A Zero-copy, CPU-efficient Serverless Architecture using Distributed Shared Memory and SmartNICs

Abstract

Cloud-native applications extensively depend on the loosely coupled microservice paradigm to take advantage of the elasticity and fine-grained resource management provided by the serverless cloud architecture. Application functions use the infrastructure layer of the serverless architecture for common functionalities like traffic management, observability, telemetry, auto-scaling, and authentication. The infrastructure layer is implemented as a service mesh that connects functions and microservices of the applications using sidecars for service mesh functionality. Therefore, the performance of the applications significantly depends on the cloud's sidecar-based infrastructure. Container-based sidecars typically rely on heavyweight kernel networking to provide layer-7 connectivity and RPC support for communication. This reliance introduces significant processing overheads and higher hardware resource consumption. In this paper, we propose ZEROCOPY-DSMS*, a novel design for the infrastructure layer of the serverless architecture.* ZEROCOPY-DSMS *enables the efficient use of SmartNICs with novel hardware accelerators and low-latency Remote Direct Memory Access (RDMA) networks for a serverless architecture based on distributed shared memory.* ZEROCOPY-DSMS *frees up CPU from heavyweight TCP/IP protocol processing and unnecessary data serialization/de-serialization tasks.*

Problem statement

Serverless computing, often referred to as Function-as-a-Service (FaaS [5]), has become a popular cloud computing service as it eases the burden on application developers to provision and manage cloud resources [12]. The "pay-as-you-go" billing and fine-grained resource elasticity of serverless computing can dramatically reduce user costs [12, 19]. By taking advantage of the microservice-based deployment paradigm, serverless computing encourages the user to convert applications into disaggregated serverless functions that are loosely coupled. This makes serverlerss computing an attractive choice for a variety of applications, *e.g.,* online web service [17], data analytics [11], etc.

Figure 1 shows the key components of the serverless data plane (we use the open-source Knative platform as a representative example, where the processing pipeline of Knative's queue proxy sidecar is shown in Figure 2). Each serverless function comprises two containers: a user container responsible for executing application logic and a separate sidecar container. All the sidecars within the cluster together form an infrastructure layer, known as a service mesh, that helps to orchestrate the loosely coupled serverless functions, integrating them into a cohesive cloud service. At the edge of this infrastructure is an ingress gateway, a cluster-wide data plane component. The ingress gateway serves as the entry point to the serverless cluster, facilitating tasks such as authentication. Additionally, a message broker, which is a "stateful" persistent component, facilitates networking between serverless functions. The broker enables users to leverage the microservices paradigm, allowing them to build complex applications by composing individual serverless functions.

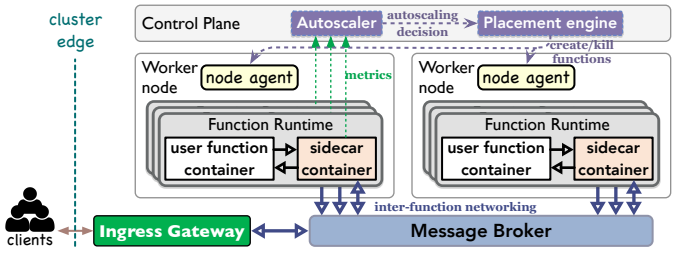

Figure 1: The architecture of serverless: data plane and control plane.

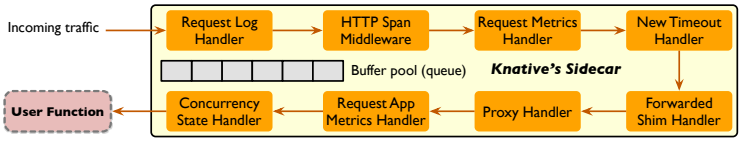

Figure 2: Processing pipeline in Knative's queue proxy sidecar. Note that we only show the ingress pipeline here.

Challenges. Despite the many advantages of serverless computing, current platforms suffer from high data plane overheads [18, 10, 6, 15], potentially contributing multiple milliseconds of delay which makes it difficult to meet the strict latency requirements of applications. This is especially true when the cloud service provider seeks to achieve the cost savings that serverless computing promises. Therefore, several key challenges need to be addressed:

- *Slow Kernel-Based Inter-Function Networking:* Kernel-based networking introduces significant performance penalties [18]. Overheads such as data copies, context switches, interrupts, protocol processing (e.g., TCP/IP) and serialization/deserialization, occur frequently when messages are moved between functions [18], resulting in milliseconds (or more) of latency.
- *Redundant Protocol Processing Within the Cluster:* Even with hardware accelerators (e.g., Intel IPU, Nvidia BlueField) with special-purpose blocks being deployed to offload or accelerate Layer 4 TCP and Layer 7 HTTP processing, serverless frameworks are still constrained by duplicate protocol processing when handling a workflow of functions at the cluster or data center level. Our approach seeks to rethink the communication model between external clients and internal functions by separating protocol stacks for external and internal communication at the cloud edge.
- *Heavyweight Service Mesh:* The loosely coupled design of the service mesh increases network latency between functions due to the additional hop (caused by kernel and user space crossings) between the user function and its respective sidecar [18, 8, 21]. This design also leads to additional CPU overhead, making the service mesh a heavyweight component [18, 21].

Our solution

We propose ZEROCOPY-DSMS, a distributed DPU-based service mesh designed to offload heavyweight sidecar proxy processing from CPUs to DPUs in a serverless infrastructure. This approach leverages the growing heterogenization of data center hardware, utilizing existing DPUs without requiring new hardware acquisitions [4]. The goal of ZEROCOPY-DSMS is to create a full-function service mesh built on distributed DPUs to enhance performance and scalability in serverless environments. The major contributions of ZEROCOPY-DSMS include: (1) A novel architecture of the data plane for the infrastructure layer of serverless computing, with modular sidecars (2) Partitioning infrastructure layer functionality and into multiple sidecar modules to leverage specialized accelerator blocks in SmartNICs (3) Eliminating redundant TCP/HTTP packet processing in the cluster (4) Distributed shared-memory based approach for efficient data transfer across functions. As shown in Figure 3, ZEROCOPY-DSMS uses a cluster-wide ingress gateway for centralized HTTP/TCP protocol processing, which strips headers and transfers payloads directly to the RDMA domain.

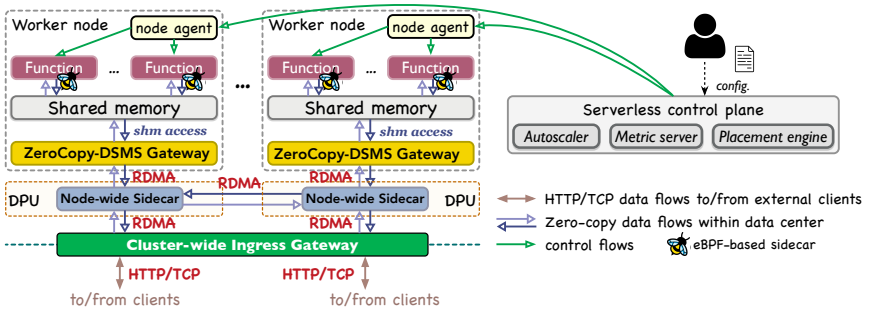

Figure 3: A architecture-level overview of ZEROCOPY-DSMS.

Partitioning Sidecar functionality. Traditional service meshes use homogeneous, monolithic sidecars, which are resource-intensive and complex to manage. ZEROCOPY-DSMS introduces a "heterogeneous" service mesh (PD-mesh) that partitions sidecar functions across software (eBPF) and hardware accelerators (DPUs). Each function is paired with an eBPF-based sidecar, while a node-wide DPU-based sidecar supports asynchronous tasks like observability, telemetry, and autoscaling. This setup allows efficient intra-node routing, redundant processing and eliminates the overhead of involving message broker (shown in Figure 2) for inter-function networking.

Intra-node shared memory data plane: For intra-node communication, ZEROCOPY-DSMS uses a zero-copy shared memory approach when interdependent functions are located on the same node. This significantly reduces resource consumption and accelerates communication compared to traditional kernel-based networking.

RDMA-based inter-node data plane: For inter-node traffic, ZEROCOPY-DSMS utilizes RDMA to maintain a zero-copy data plane across nodes, supporting RDMA over converged Ethernet (RoCE) to ensure broad applicability in diverse data center environments. A lightweight per-node agent interfaces inter-node traffic with the intra-node data plane. We use a lightweight per-node agent on the host to seamlessly interface inter-node traffic with the intra-node data plane.

Approach for Implementation

ZEROCOPY-DSMS architecture involves multiple components and sub-systems. We have further outlined the detailed system in the extended version [1]. We are building a new module in NGINX [9] to implement the cluster-wide ingress gateway and perform protocol stack translation. We using DOCA APIs and BlueFiled DPUs to enable efficient RDMA communication while optimizing the usage of scarce hardware resources.

Competitive approaches

There has been many efforts focused on streamlining service mesh. SPRIGHT [18] uses lightweight eBPF-based sidecars to replace heavyweight container-based sidecars in the service mesh data plane. However, eBPF-based sidecars cannot support full-functional service mesh processing (particular in L7) such as TLS termination, connection splicing. Overall, these approaches only consider *a point solution* in the design space, while ZEROCOPY-DSMS fully considers the heterogeneous data plane components and maximizes their advantages. There exists previous work focusing on using RDMA-based RPC to improve data center RPC. FaSST [14], RFP [20], and eRPC [13] achieve high performance with RDMA but still require migration of applications. Fuyao [16] uses DPU to enable direct function communication in the serverless data plane. However, Fuyao [16] sets a sidecar for each function instance, which has proved to be inefficient. Further, Fuyao [16] does not offer isolation or mediated transfer between functions when needed, e.g., when functions are from different security domains. It only supports IPC (with/without intra-node engine) between functions.

Current status and next steps

ZEROCOPY-DSMS, even with a naive offloading approach, reduces host CPU utilization with minimal performance degradation. Utilizing L2/L3 acceleration on the DPU can further enhance ZEROCOPY-DSMS's performance, even when using less powerful DPU cores. Additionally, techniques such as targeted kernel performance acceleration and optimized IRQ handling can significantly boost ZEROCOPY-DSMS's efficiency. Our prototype is being implemented and evaluated on a testbed featuring Mellanox Bluefield-2 and Bluefield-3 DPUs. To further improve the ZEROCOPY-DSMS platform, we plan to incorporate support for heterogeneous data processing units (DPUs), enabling workload resource management to be vendor-agnostic. As various specialized devices, including DPUs [3, 7], are increasingly adopted in cloud infrastructures, ZEROCOPY-DSMS can be seamlessly deployed in environments like the GreenLake cloud infrastructure, allowing providers to utilize existing DPUs without the need for new hardware acquisitions. At TechCon, we aim to showcase the integration of ZEROCOPY-DSMS with the GreenLake [2] platform.

References

[1] Extended version of the paper. Link to the Extended version of the paper, 2020. [EXTENDED VERSION OF THE PAPER].

[2] HPE Greenlake. Link, 2020. [ONLINE].

[3] Pensando introduces Distributed Services Platform available through HPE GreenLake and HPE infrastructure solutions. Link, 2020. [ONLINE].

[4] NVIDIA BlueField Networking Platform. `https://www.nvidia.com/en-us/networking/products/data-processing-unit/`, 2024. [ONLINE].

[5] What is FaaS? `https://www.ibm.com/topics/faas`, 2024. [ONLINE].

[6] Sol Boucher, Anuj Kalia, David G. Andersen, and Michael Kaminsky. Putting the "micro" back in microservice. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 645–650, Boston, MA, July 2018. USENIX Association.

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

[8] Jingrong Chen, Yongji Wu, Shihan Lin, Yechen Xu, Xinhao Kong, Thomas Anderson, Matthew Lentz, Xiaowei Yang, and Danyang Zhuo. Remote procedure call as a managed system service. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 141–159, Boston, MA, April 2023. USENIX Association.

[9] F5 Networks, Inc. NGINX: Advanced Load Balancer, Web Server, & Reverse Proxy. `https://www.nginx.com/`, 2022. [ONLINE].

[10] Zhipeng Jia and Emmett Witchel. Nightcore: Efficient and scalable serverless computing for latency-sensitive, interactive microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, page 152–166, New York, NY, USA, 2021. Association for Computing Machinery.

[11] Chao Jin, Zili Zhang, Xingyu Xiang, Songyun Zou, Gang Huang, Xuanzhe Liu, and Xin Jin. Ditto: Efficient serverless analytics with elastic parallelism. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 406–419, New York, NY, USA, 2023. Association for Computing Machinery.

[12] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A. Patterson. Cloud programming simplified: A berkeley view on serverless computing, 2019.

[13] Anuj Kalia, Michael Kaminsky, and David Andersen. Datacenter RPCs can be general and fast. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 1–16, Boston, MA, February 2019. USENIX Association.

[14] Anuj Kalia, Michael Kaminsky, and David G. Andersen. FaSST: Fast, scalable and simple distributed transactions with Two-Sided (RDMA) datagram RPCs. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 185–201, Savannah, GA, November 2016. USENIX Association.

[15] Collin Lee and John Ousterhout. Granular computing. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '19, page 149–154, New York, NY, USA, 2019. Association for Computing Machinery.

[16] Guowei Liu, Laiping Zhao, Yiming Li, Zhaolin Duan, Sheng Chen, Yitao Hu, Zhiyuan Su, and Wenyu Qu. Fuyao: Dpu-enabled direct data transfer for serverless computing. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 431–447, New York, NY, USA, 2024. Association for Computing Machinery.

[17] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. Characterizing microservice dependency and performance: Alibaba trace analysis. SoCC '21, page 412–426, New York, NY, USA, 2021. Association for Computing Machinery.

[18] Shixiong Qi, Leslie Monis, Ziteng Zeng, Ian-chin Wang, and K. K. Ramakrishnan. Spright: Extracting the server from serverless computing! high-performance ebpf-based event-driven, shared-memory processing. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 780–794, New York, NY, USA, 2022. Association for Computing Machinery.

[19] Alireza Sahraei, Soteris Demetriou, Amirali Sobhgol, Haoran Zhang, Abhigna Nagaraja, Neeraj Pathak, Girish Joshi, Carla Souza, Bo Huang, Wyatt Cook, Andrii Golovei, Pradeep Venkat, Andrew Mcfague, Dimitrios Skarlatos, Vipul Patel, Ravinder Thind, Ernesto Gonzalez, Yun Jin, and Chunqiang Tang. Xfaas: Hyperscale and low cost serverless functions at meta. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 231–246, New York, NY, USA, 2023. Association for Computing Machinery.

[20] Maomeng Su, Mingxing Zhang, Kang Chen, Zhenyu Guo, and Yongwei Wu. Rfp: When rpc is faster than server-bypass with rdma. In *Proceedings of the Twelfth European Conference on Computer Systems*, EuroSys '17, page 1–15, New York, NY, USA, 2017. Association for Computing Machinery.

[21] Xiangfeng Zhu, Guozhen She, Bowen Xue, Yu Zhang, Yongsu Zhang, Xuan Kelvin Zou, XiongChun Duan, Peng He, Arvind Krishnamurthy, Matthew Lentz, Danyang Zhuo, and Ratul Mahajan. Dissecting overheads of service mesh sidecars. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*, SoCC '23, page 142–157, New York, NY, USA, 2023. Association for Computing Machinery.