Towards Offloadable and Migratable Microservices on Disaggregated Architectures: Vision, Challenges, and Research Roadmap

Xiaoyi Lu and Arjun Kashyap
University of California, Merced
xiaoyi.lu@ucmerced.edu, akashyap5@ucmerced.edu

Abstract

Microservice and serverless computing systems open up massive versatility and opportunity to distributed and datacenter-scale computing. In the meantime, the deployments of modern datacenter resources are moving to disaggregated architectures. With the flourishing growths from both sides, we believe this is high time to write this vision paper to propose a potential research agenda to achieve efficient deployments, management, and executions of next-generation microservices on top of the emerging disaggregated datacenter architectures. In particular, we envision a critical systems research direction of designing and developing offloadable and migratable microservices on disaggregated architectures. With this vision, we have surveyed the recent related work to demonstrate the importance and necessity of researching it. We also outline the fundamental challenges that distributed systems and datacenter-scale computing research may encounter. We further propose a research roadmap to achieve our envisioned objectives in a promising way. Within the roadmap, we identify potential techniques and methods that can be leveraged.

1. Introduction

Microservices and serverless computing are becoming the new paradigm in cloud computing. The microservice architecture of decoupling monolithic applications to loosely-coupled microservices is gaining traction in large datacenters due to the ability to meet strict performance and availability constraints. Big cloud providers have already started adopting the microservice application model [5, 8, 9]. Similarly, many public cloud platforms are offering serverless computing services like AWS Lambda [2], Azure Functions [3], and Google Cloud Functions [6] as the platform automatically manages the infrastructure and server to run these serverless functions.

With the increasing popularity of microservices, current efforts have been put to study the characteristics, requirements, and implications they have on the cloud system stack [17, 30, 43, 46, 51]. Due to the strict service level objectives (Quality-of-Service requirements) by customers for throughput and tail latency, microservices put pressure on cloud architecture ranging from hardware to operating system and all the way up to application design. Lastly, datacenters have a huge energy footprint [33] and these days the cloud providers equally focus on both performance and energy-efficiency of the applications to reduce the carbon footprint.

On the other hand, Disaggregated Architecture for modern datacenters becomes a hot research topic recently. Disaggregated architecture splits the existing monolithic datacenter tier into at least two independent resource pools (i.e., compute pool and storage pool) that communicate over fast interconnects, such as high-speed Ethernet or InfiniBand. By sharing the storage pool across diverse datacenter services, disaggregated architecture consolidates computational and storage resources independently. Thus, it allows independent optimization and resource scaling, and reduces total cost. The cost and efficiency benefits of disaggregated architecture have attracted interest from both academia and industry [26, 16, 31, 44, 41]. Modern public cloud providers such as Amazon and Microsoft Azure Cloud [13] have already adopted high-level compute and storage disaggregation.

With these trends, more and more microservices or serverless computing-based applications will be deployed and run on top of disaggregated architectures of modern and next-generation datacenters [38]. In this kind of deployments, we envision that next-generation microservices will be more often offloaded and executed on remote computational devices or disaggregated resources. In the meantime, these microservices need to be migratable across different layers of heterogeneous hardware platforms, due to the requirements of manageability, performance, load balancing, and so on. In other words, we believe that offloadable and migratable microservices will become a key computing paradigm on disaggregated datacenter architectures in the future.

In this paper, we first present a brief survey on related work (Section 2) to give a high-level overview of existing studies in related research directions. Then, we outline a vision (Section 3) for demonstrating how and why microservices should be offloaded and migratable on disaggregated datacenter architectures. The associated challenges that distributed systems research may encounter are discussed in Section 4. To address these fundamental challenges, we propose a roadmap (Section 5), mapping key research challenges to bodies of work that have the potential to realize high-performance, scalable, energy-efficient, and QoS-aware offloadable and migratable microservice systems on disaggregated architectures. The key contributions of this paper are:

- Envisioned a critical systems research direction of designing and developing next-generation offloadable and migratable microservices on emerging disaggregated datacenter archi-
• Surveyed the recent related studies in the literature and the community to demonstrate the importance and necessity of this research direction.
• Outlined a vision and the associated fundamental challenges that distributed systems and datacenter-scale computing research may encounter when building next-generation off-loadable and migratable microservices.
• Proposed a research roadmap to achieve our envisioned objectives in a promising way. Within the roadmap, we identify potential techniques and methods that can be leveraged.

2. A Brief Survey on Related Work

This section presents a brief survey on recent studies, which are closely related to our research agenda.

2.1. Microservice Benchmarking

Research has been done on evaluating and benchmarking microservices [17, 30, 43, 46, 51, 19] but none of them discuss the pros and cons when making microservices offloadable and migratable. DeathStarBench [17] contains microservices that are representative of large end-to-end services and studies their implications across the cloud system stack. It also shows that running microservices on low power cores leads to slight performance degradation but saves power. Thus, these cores could be utilized for microservices that are off the critical path. μSuite [43] investigates the impact of network and OS overheads on microservice median and tail latency. Ueda et al. [46] present the impact of language runtime and hardware architecture on microservice performance. Zhou et al. [51] identify the gaps between existing benchmark systems and industrial microservice systems and develop a new microservice benchmarking system. ppbench [30] proposes a benchmark to study the performance impact of programming languages and networks on microservices.

2.2. Disaggregated Datacenter Architecture

Disaggregated datacenter architecture has many advantages like flexibility and independent scaling of resources (compute, network, and/or storage), fine-grained resource provisioning, and higher utilization. We believe this architecture will continue to evolve and include new computational devices which microservices should benefit from. A lot of work has been done in the community to allow applications to reap the benefits of resource disaggregation [11, 42, 41, 35, 32, 31]. LegoOS [41] is a new OS model for hardware resource disaggregation that splits traditional operating system functionalities into loosely-coupled units. soNUMA [35] enables low latency and distributed in-memory processing by eliminating data movement across network stack and PCIe interface. Legtchenko et al. [31] discuss the concept of disaggregated storage at rack-level and propose a novel storage fabric for different disaggregation types for cloud storage.

2.3. Near-Data Processing

With the recent trends in disaggregated datacenter architectures and pushing towards improving energy-efficiency, many applications are being re-designed to move compute near to memory/storage. Near-Data Processing (NDP) has been explored in the areas of databases [12, 14, 25, 36], computer vision [24], machine-learning inference [49], big-data systems [20], and key-value stores [15, 29, 47] with the vision to reduce data movement. We believe microservices should also exploit this paradigm. Hu et al. [24] present a storage system that adjusts the dataset resolution within the storage device by utilizing idle SSD cores that are otherwise used by flash translation layer and garbage collection inside an SSD. E3 [33] is a microservice execution platform that off-loads microservices to low-power SmartNIC processors to improve energy-efficiency without much loss in latency. E3 also migrates microservices to host to eliminate SmartNIC overloading via a communication-aware service placement algorithm. Though some of our motivations are similar to prior work, e.g., reducing energy consumption on heterogeneous compute substrates, we not only want microservices to be migratable but also to benefit from NDP by leveraging computational devices in future datacenter architectures without losing on the productivity of developers.

2.4. Resource Virtualization & Management

Microservices are generally built on top of resource virtualization techniques like containers and virtual machines (VMs). Apache OpenWhisk [1] is an open-source, distributed serverless platform that executes functions in response to events at any scale. OpenWhisk manages the infrastructure, servers, and scaling using Docker containers. It supports many container frameworks and programming languages in which one can write functional logic and run them in response to events or from HTTP requests.

I/O virtualization introduces a major bottleneck for high-performance networks. Remote Direct Memory Access (RDMA) virtualization is one way to achieve migration of microservices in the new datacenter architecture across diverse compute units. Current work in this space, namely, SR-IOV [37, 27], LITE [45], vSocket [48], HyV [39], and FreeFlow [28] have heavy connection setup costs and may not be suitable for container-based virtualized environments. MassQ [23] is a software-defined RDMA virtualization technique to allow VMs in public clouds to realize virtual RoCE devices. MigrOS [40] allows transparent live migration for RDMA-enabled containers.

3. Vision

As shown in Figure 1, we envision the next-generation disaggregated datacenter architectures to comprise of persistent memory [50, 22] (PMEM), computational PMEM
As indicated earlier, the availability of computational devices near storage and memory, like CSDs and CPM, and next-generation datacenter platforms provide novel opportunities to co-design microservices/serverless applications and their underlying runtime to significantly improve energy-efficiency, performance, scalability, QoS, and so on. However, much of the work to utilize computational devices is still application-specific [47, 14, 7] and cannot be used by general-purpose microservices or serverless functions.

Data-intensive applications spend a significant amount of time and energy to move data towards compute. But with the availability of compute units near the memory and storage devices, certain tasks can be offloaded to these devices to save energy consumption and improve the performance of the application. Thus, it becomes critical to co-design microservice applications to take advantage of the next-generation datacenter hardware to reduce data movement and lessen their carbon footprint. Based on these trends, we envision that hardware evolution and cross-layer software co-design will continue to be the two most important driving forces for the architectural evolution and performance-boosting of future serverless computing systems.

Based on these observations, we identify several fundamen-
eral research challenges to achieve the goals of offloadable and migratable microservices/serverless functions, which aim to fully utilize the provided resources on modern and next-generation datacenter infrastructures.

1. How can we design effective and efficient runtime systems for microservices and serverless functions that can take advantage of the novel features of next-generation datacenter hardware?

2. How can we co-design with applications and microservice platforms to achieve near-native performance while reducing energy consumption?

3. How to offload and migrate the entire microservice or a portion of it to computational devices in an energy-efficient and QoS-aware way without burdening any of resource-constrained devices?

4. How to offload such tasks with minimal code changes to the applications to keep the developer productivity high?

5. What kind of benefits can be achieved through these new designs? How to benchmark these designs with microservice platforms and applications?

5. Research Roadmap

This section presents our proposed research roadmap as shown in Figure 2.

5.1. Standardized Programmable Hardware Interface

First of all, it is a challenging task to offload an entire or a portion of a microservice to heterogeneous compute units in the datacenter. This is because we need a standardized, generic, and programmable interface from the underlying hardware layer. Creating such an interface is challenging in a disaggregated datacenter architecture as the heterogeneous computational devices could be interconnected via the memory bus, intra-node, or inter-node interconnects (like PCIe, CXL, Gen-Z, NVLink, RoCE, InfiniBand, etc.). Interacting with these devices typically needs to change systems or applications a lot currently. Hence, future research in this space should target offloading any functionality across the datacenter in a way that has minimal impact on the productivity of the developer and supports both built-in and resource-conservative user-defined functions. Even though this is a difficult research task, we do see success stories in the community. For instance, RDMA with verbs and NVMe protocols are becoming standardized programmable interfaces for high-speed networks and storage devices. We envision that more standardized programmable interfaces will become available in the future.

5.2. Portable Compute Abstractions

When migrating/offloading microservices on various compute devices in the datacenter, the ideal technique would be not to rewrite/recompile code for each device. For instance, it may be a difficult problem to avoid the cross-compilation issue when running microservices on different compute devices. One possible solution is we could consider generating a number of binaries and using the correct one on each compute device.

In addition, producing new compute abstractions would help us in reaching this goal. Such abstractions should minimize the compute footprint in order to reduce the compute movement cost across the datacenter and allow microservices to transfer/share the output of offloaded compute. Thus, the community should propose new abstractions which not only increase productivity but also maintain portability and performance. So far, the most common ways to deploy microservices are based on virtual machines or containers. To solve all the challenges we listed earlier, it may be an interesting and open challenge to rethink what could be the best way to design portable compute abstractions for microservices.

5.3. Offload-aware Systems, Runtimes, and APIs

Microservices could run numerous kinds of workloads, which can be both stateful and stateless workloads. We foresee different approaches should be adopted to handle stateful and stateless microservices. In order to fully support various workloads and execute offloaded functions on different kinds of computational devices, we need offload-aware systems, runtimes, and APIs. Developing such kind of systems to fully leverage diverse compute units in an application-agnostic manner and at the same time add minimal overhead is an interesting research direction. For example, an offload-aware communication and I/O runtime should be able to automatically and transparently select the best protocols for microservice applications based on the capabilities of the devices where the microservices are offloaded to.

5.4. Live Migration Capability

Microservice migration, especially for stateful microservices, is important to avoid overloading and achieve good utilization across computational devices in a datacenter. Migration needs not to be just from one host/server to another. It could be, for example, from a local computational storage device to a remote computational persistent memory device or the other way around. We need to develop new mechanisms that enable fast live migration of microservices using different network protocols and interconnects while preserving performance. Another important aspect of enabling live migration would be to figure out which heterogeneous compute unit has the resources to support the migrating microservice without violating the QoS support.

5.5. Task-Aware Near Data Offload

For maximum utilization of available compute resources and to reduce data movement and energy consumption, task and capability aware offloading of microservices is necessary. A good offload strategy should decide where to offload and also take into consideration offloading costs vs. rewards. Future research in task-aware offloading should explore designing new metrics and models which could characterize and pre-
dict when and on which computational device it would be useful to offload microservices in disaggregated datacenters. The potential modeling and decision-making approaches can be built with some artificial intelligence techniques, such as reinforcement learning.

5.6. Dynamic Resource Partitioning

Due to the heterogeneous nature of compute, memory, and workloads on a computational device in a datacenter, dynamic resource partitioning for offloaded functions is a challenge. One must ensure that running microservices or functions on CSDs/CPMs should not impact the performance of their main task – I/O processing. To achieve this, hardware vendors may need to provide more resources or mechanisms on hardware devices to help systems designs. For example, dynamic I/O queues or resource pools can be provided from the hardware layer to achieve better designs.

5.7. Isolation and Load Balancing

As these computational devices will run multiple microservices/offloaded functions, there should be strong security and isolation guarantees among them as present on a host server. Some possible technology examples include hardware-based data encryption, namespaces, protection domains, etc.

Also, the load balancer which would analyze the load of all or nearby devices should make offloading decisions based on data locality and device load. To achieve scalable load balancing, both centralized and decentralized load balancing algorithms should be designed.

5.8. Energy and QoS Aware Systems

All microservices have to meet their Service Level Objectives (SLOs), so offloading of microservices, especially compute-intensive in nature, should occur if running them on computational devices does not violate their QoS guarantees. Another important factor to consider while executing microservices is to respect the pre-defined energy/power budget. Hence, the challenge would be to balance the performance-energy trade-off based on varying application and user needs. These goals can be achieved through accurate modeling and enforcement in microservice systems. Artificial intelligence techniques may again play a very important role in this research direction.

5.9. Co-design Hardware & Software

Multiple layers of software frameworks, programming model semantics, and newer architectures have rendered the design process to be complicated and isolated. Thus, a holistic cross-layer approach is critical to tackling the challenges at these layers. The community needs to develop runtimes or systems that could provide efficient virtualization mechanisms, live migration capability, and support all the other research challenges outlined above. Then microservices and serverless functions could be co-designed to coordinate and interact with the underlying runtimes and reap all the benefits of next-generation datacenter infrastructures.

We envision the next round of potential innovation to happen in another 5-10 years after materializing the research roadmap discussed here. This research roadmap needs a lot of research work and innovations from the community to achieve these goals.

6. Conclusion

This vision paper emphasized an important systems research direction on offloadable and migratable microservices on disaggregated datacenter architectures. We performed a systematic survey on related studies, presented our vision and associated research challenges, and proposed a research roadmap to achieve the envisioned objectives. In the future, we plan to build a new microservice or serverless computing system, which can efficiently support the features outlined in our research agenda. We also wish our vision paper can bring more attention from the community to collaborate and participate in this research direction.

Acknowledgement

We would like to thank Stanko Novakovic and anonymous reviewers for their valuable feedback. This work was supported in part by NSF research grant CCF #1822987.
References

[1] Apache OpenWhisk. https://openwhisk.apache.org/.
[2] AWS Lambda - Serverless Compute. https://aws.amazon.com/lambda/.
[3] Azure Functions Serverless Architecture. https://azure.microsoft.com/en-us/services/functions/.
[4] Computational SSDs. https://www.niaa.org/sites/default/files/ISECSchema/2019/Presentations/Computational_SSDs_Final.pdf.
[5] Decomposing Twitter: Adventures in Service-Oriented Architecture: Efficiently Support Analytical Workloads on Cloud-Native Twitter-Adventures in Service-oriented Architecture.
[6] Google Cloud Functions. https://cloud.google.com/functions/.
[7] Memory-Centric Active Storage. https://ibm.github.io/mcas/.
[8] Microservices Workshop: Why, What, and How to Get There. https://www.slideshare.net/adriancockcroft/microservices-workshop-craft-conference.
[9] The Evolution of Microservices. https://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference.
[10] Marcos K. Aguilara, Nadav Ami, Irina Calciui, Xavier Deguilllard, Jayneel Gandhi, Stanco Novakovic, Arun Ramanathan, Pratap Subrahmanyan, Lalith Suresh, Kuran Tati, Rajesh Venkataabramaniam, and Michael Wei. Remote Regions: A Simple Abstraction for Remote Memory. In 2018 USENIX Annual Technical Conference (USENIX ATC 18), pages 775–787, Boston, MA, July 2018. USENIX Association.
[11] Krste Asanovic. FireBox: A Hardware Building Block for 2020 Warehouse-Scale Computers. FAST ’14, Santa Clara, CA, February 2014. USENIX Association.
[12] Oreoluwanmiwa O. Babarinsa and Stratos Idreos. JAFAR: Near-Data Processing for Databases. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD ’15, page 2069–2070, New York, NY, USA, 2015. Association for Computing Machinery.
[13] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nikalant, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jisheng Wu, Huseyn Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Luna, and Onur Mutlu. Processing-In-Memory: A Workload-Driven Architecture with Strong Consistency. In Proceedings of the 23rd Annual International Symposium on Computer Architecture, ISCA ’16, page 153–165. IEEE Press, 2016.
[14] Boncheol Gu, Andre S. Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Moo Sang Kwon, Chanho Yoo, Sangyeun Cho, Jueheon Jeong, and Duckhyun Chang. Biscuit: A Framework for Near-Data Processing of Big Data Workloads. In Proceedings of the 43rd International Symposium on Computer Architecture, ISCA ’16, page 153–165. IEEE Press, 2016.
[15] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. Efficient Memory Disaggregation with InfiniSwap. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pages 649–667, Boston, MA, March 2017. USENIX Association.
[16] Martin Grambow, Lukas Meusel, Erik Wittern, and David Bermbach. Benchmarking Microservice Performance: A Pattern-based Approach. In Proceedings of the 22nd Annual ACM Symposium on Applied Computing, pages 232–241, 2020.
[17] Shashank Gugnani, Arjun Kashyap, and Xiaoyi Lu. Understanding the Idiosyncrasies of Real Persistent Memory. Proceedings of the VLDB Endowment, 14(4):626–639, 2021.
[18] Zhiqiang He, Dongyang Wang, Bintzhang Fu, Kun Tan, Bei Huai, Zhi-Li Zhang, and Kai Zheng. MaQ: RDMA for Virtual Private Cloud. In Proceedings of the Annual Conference of the ACM on Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM 20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.
[19] Yu-Ching Hu, Murtuza Taher Lokhandwala, Te L., and Hung-Wei Tseng. Dynamic Multi-Resolution Data Storage. In Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO ’19, pages 196–210, New York, NY, USA, 2019. Association for Computing Machinery.
[20] A.R. Hurson, L.L. Miller, S.H. Pakzad, M.H. Eich, and B. Shirazi. Parallel Architectures for Database Systems. volume 28 of Advances in Computers, pages 107 – 151. Elsevier, 1988.
[21] Yuan He, Boncheol Gu, Andre S. Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael J. Jordan, and I. Mark Aisen. A Distributed Framework for Emerging AI Applications. In Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation, OSDI’18, page 561–577, USA, 2018. USENIX Association.
[35] Stanko Novakovic, Alexandros Daglis, Edouard Bugnion, Babak Falsafi, and Boris Grot. Scale-out NUMA. In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS ’14, page 3–18, New York, NY, USA, 2014. Association for Computing Machinery.

[36] Kwanghyun Park, Yang-Suk Kee, Jignesh M Patel, Jaeyoung Do, Chaniik Park, and David J Dewitt. Query Processing on Smart SSDs. IEEE Data Eng. Bull., 37(2):19–26, 2014.

[37] PCI-SIG. Single Root I/O Virtualization. https://pcisig.com/specifications/iov/singleroot/.

[38] Nathan Pemberton. Exploring the Disaggregated Memory Interface Design Space. http://word19.ece.cornell.edu/word19-paper8-camera.pdf.

[39] Jonas Pfefferle, Patrick Stuedi, Animesh Trivedi, Bernard Metzler, Ionnis Koltsidas, and Thomas R. Gross. A Hybrid I/O Virtualization Framework for RDMA-Capable Network Interfaces. In Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE ’15, page 17–30, New York, NY, USA, 2015. Association for Computing Machinery.

[40] Maksym Planeta, Jan Bierbaum, Leo Sahaya Daphne Antony, Torsten Hoefler, and Hermann Härtig. MigrOS: Transparent Operating Systems Live Migration Support for Containerised RDMA-applications. https://arxiv.org/abs/2009.06988, 2020.

[41] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiyi Zhang. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pages 69–87, Carlsbad, CA, October 2018. USENIX Association.

[42] Tobias Vinçon, Arthur Bernhardt, Ilia Petrov, Lukas Weber, and Andreas Koch. NKV: Near-Data Processing with KV-Stores on Native Computational Storage. In Proceedings of the 16th International Workshop on Data Management on New Hardware, DaMoN ’20, New York, NY, USA, 2020. Association for Computing Machinery.

[43] Jian Yang, Juno Kim, Morteza Hoseinzadeh, Joseph Izraelevitz, and Steve Swanson. An Empirical Guide to the Behavior and Use of Scalable Persistent Memory. In 18th USENIX Conference on File and Storage Technologies (FAST 20), pages 169–182, Santa Clara, CA, February 2020. USENIX Association.

[44] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chenjie Xu, Chao Ji, and Wenyun Zhao. Benchmarking Microservice Systems for Software Engineering Research. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, ICSE ’18, page 323–324, New York, NY, USA, 2018. Association for Computing Machinery.