The Serverless Data Center: Hardware Disaggregation Meets Serverless Computing

Nathan Pemberton, Johann Schleier-Smith
UC Berkeley

Abstract

Serverless computing and hardware disaggregation are recent movements that have progressed largely independently, even though they share a common goal: to shape the future of cloud computing. Serverless computing is a software layer that simplifies cloud programming, even without changes to the underlying hardware, whereas hardware disaggregation challenges the established approach to data center architecture, breaking up traditional server units into their individual components and rewiring them in alternative ways. While a superficial similarity is apparent, the two movements are synergistic in fundamental technical ways. Cross-pollination of ideas promises advances on both sides, and co-design could be essential and rewarding in the long run.

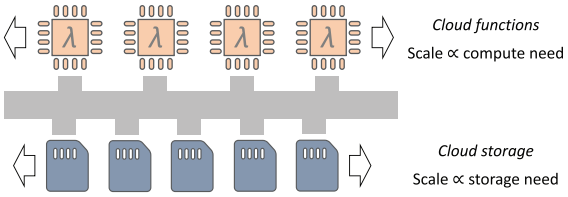
1. Introduction

Serverless computing arose to meet demand from application developers for a simplified cloud programming model [1]. It provides a general-purpose computing abstraction, *cloud functions*, that lets programmers deploy code written in a high-level language, specify events as execution triggers, and pay only for actual resource consumption, typically CPU and memory metered in 100 ms increments. The cloud provider ensures elastic scaling, provisioning resources as needed by drawing from pools shared across many customers. Serverless computing complements cloud functions with stateful services, such as object storage, that similarly bill according to consumption (e.g., space used $\times$ time) rather than according to allocation. The core problem that serverless computing fixes, the thing that makes traditional cloud programming difficult, is the mismatch between the granularity of resources such as VMs and disks, and the natural granularity of the application software and workloads.

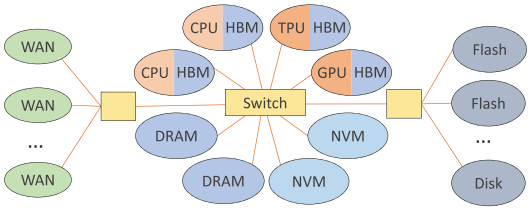
Today one can argue, with good reason, that the term "serverless computing" is a misnomer as the underlying hardware infrastructure remains traditional servers. Hardware disaggregation seems to answer this objection, presenting a vision of repackaging the components of servers, breaking down the traditional boundaries and offering shared resource pools of memory, disks, and accelerators, all linked through a high-performance interconnect [2, 3]. It also provides the flexibility to deploy the right mix of resources and to evolve this mix over time to meet shifts in demand. Figure 1 compares the architecture of a serverless application to the architecture of a data center with disaggregated hardware.

The connection between serverless computing and hardware disaggregation is more than superficial. Today's serverless computing represents the extreme progression of a trend towards *logical disaggregation* of software in which developers split applications into components, each specialized to a purpose, running and scaling independently, and having its own resource profile. This trend goes back at least to the origin of Service-Oriented Architecture (SOA) [4] and was furthered by the microservices movement [5]. In practice logical disaggregation leads to physically distributed applications since component programs may be deployed on separate servers. In a modern cloud application, internal fan-out can involve scores of servers for processing just one web request [6]. Cloud functions encourage further decomposition of applications into independent pieces, in practice increasing the number of servers involved in any given unit of work even more.

Imagine illuminating each bit of hardware touched by a



(a) Serverless computing



(b) Hardware disaggregation

Figure 1: (a) Serverless cloud functions and shared storage can be used to build elastic applications. The serverless runtime creates and destroys cloud function instances in response to demand. The storage layer, e.g., object storage or key-value storage, scales independently. (b) With hardware disaggregation, hardware resources become first-class citizens on the network. Compute nodes take the form of System in Package (SiP) modules that couple CPUs, network interfaces, and a small amount of high-speed memory (HBM). The design allows for flexible scaling and allocation of resources.

request in a modern cloud application—the resulting constellation makes it abundantly clear that today's cloud applications *already are disaggregated*, with serverless applications exhibiting especially great dispersion and high intricacy. The greatest fears and objections to hardware disaggregation, how to program for it and how its performance profile impacts applications, now appear unfounded. Programmers are already writing their applications to use fine-grained, distributed resources, and advances in disaggregated hardware will only improve performance and usability.

In our vision, the future cloud data center combines hardware disaggregation with serverless computing, creating a co-designed system that we call the *serverless data center*. While this may take time to emerge, there are immediate opportunities for ideas from hardware disaggregation to inform the progression of serverless computing, as well as the other way around. For example, serverless platform developers can have confidence that access to shared remote data will get much faster, whereas hardware developers can use the isolation models of serverless computing as guidance when developing security primitives.

In Section 2, we explain how logical and physical disaggregation are both fundamental consequences of the advantages serverless computing provides, and expand on the synergy with hardware disaggregation. Section 3 surveys opportunities for cross-pollinating research, and Section 4 describes our approach to serverless hardware/software co-design. We summarize key takeaways in Section 5.

2. Serverless Computing and Disaggregation

2.1. Serverless Computing

Serverless computing is a rapidly developing field seeing both improvements in commercial offerings and new academic proposals [7–9]. The defining characteristics of serverless platforms are elasticity and pay-per-use (see Figure 2), characteristics that lead inevitably to scheduling resources in disparate parts of the data center. To see why this is, consider the challenge of providing the sort of rapid scaling possible in mature serverless environments such as AWS Lambda or Google App Engine, which can provide hundreds or even thousands of cores within seconds, and can reclaim them quickly when idle. While it is possible to schedule 100 cores within a server, or 1,000 cores within a rack, it is much faster and more efficient to find available resources when drawing from the data center as a whole, or from a large portion of it. Adding more dimensions to the resource allocation problem, e.g., specifying varying amounts of memory along with compute, poses an even greater bin-packing challenge and drives further physical distribution. Not all serverless workloads require thousands of compute nodes, but efficient resource utilization requires that resource allocations track demand with high fidelity. Again, finding available resources gets easier, and faster, when the pool is larger.

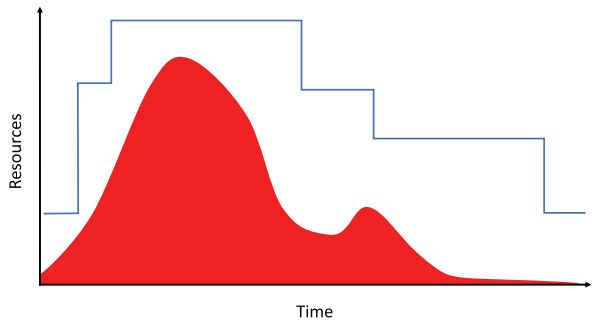


Figure 2: Serverless computing users pay for the resources consumed by their workload (red area). Traditional cloud VM users pay for resource capacity, and since autoscaling techniques can be slow to respond they must reserve and pay for extra capacity in anticipation of load spikes (blue line).

While trends toward application distribution (via logical disaggregation) have been underway for some time, so long as cloud providers offered only a VM-based computing model, hardware disaggregation risked a mismatch to server-oriented customer expectations. Serverless computing changes this calculation dramatically. By decomposing an application into cloud functions, programmers already think about maintaining locality where it matters, as well as hiding latency where it could hurt (e.g., by using batching or parallel dispatch). Also, because programmers can define new serverless functions much more easily than they can create server-based services, they often break down their applications in a very fine-grained way.

We summarize how serverless computing supports hardware disaggregation and the future serverless data center as follows:

- Furthering the logical disaggregation of cloud applications as a consequence of providing elasticity and efficient multi-tenant resource sharing.
- Encouraging the fine-grained decomposition of applications by making it easy to deploy collections of cloud functions.
- Raising the level of abstraction of cloud programming, removing the expectation of servers and giving the cloud provider more latitude for far-reaching redesign.
- Providing an evolutionary path for software, one compatible with current data center hardware and able to take advantage of future hardware.

2.2. Disaggregated Hardware

Independently of the emergence of serverless cloud computing, researchers from the architecture and operating system communities have proposed the concept of hardware disaggregation [2, 3, 10–14]. With hardware disaggregation, each resource type (including compute, memory, and storage) becomes a first-class citizen over the network, either within a rack or across the data center. In this paper, we will focus on a hierarchical model of disaggregation where resources are accessible across the data center, with potentially varying

performance based on locality. This model is enabled today by the convergence of off-package memory and network bandwidth trends [15, 16]. New network [17, 18] and memory [19] technologies, coupled with novel hardware-accelerated interfaces [20–23], will make it even more compelling.

The case for disaggregated hardware shares some motivation with serverless computing: resource utilization in the data center can be very low due to over-provisioning and inflexible allocations [24], and the optimal hardware mix must evolve over time to meet changing workloads [25–27].

While serverless computing makes a transition to hardware disaggregation natural, it also will benefit more from advances in disaggregation than does software designed for serverful computing. Among other things, the tightly-coupled networks and rich interface semantics provided by disaggregated hardware will alleviate the network interface bottleneck experienced by today's serverless applications [28].

2.3. Shared Limitations

Serverless computing is a boon to an increasing number of applications, but there are many applications that today are better served by traditional VMs [1, 28]. In some cases disaggregated hardware can help, but when limitations encountered are fundamental consequences of locality both serverless computing and disaggregation may struggle with the workload.

In big data processing, operations such as broadcast and shuffle involve more network operations when compute resources are widely distributed than when they are co-located, as co-location allows combining data before sending or sharing one copy after receiving. Disaggregated hardware can make communication much cheaper, but does not eliminate locality effects or the memory hierarchy. Perhaps the traditional server-based locality unit can be relaxed, but placement decisions are called for at some level of scale.

Coordination-intensive applications such as OLTP databases also are difficult to provide on top of today's cloud functions, which lack shared memory and limit communication in ways that preclude use of traditional distributed systems algorithms. Some of these constraints may be inherent to elastic resource provisioning, suggesting that disaggregated hardware might be of little help. When cloud functions run into limitations, serverless computing complements them with application-specific services, also known as Backend as a Service (BaaS), built using traditional server primitives [1]. In the case of databases, significant gains can be realized from specialized servers [29].

This suggests that a data center built with disaggregated hardware should also include a certain amount of traditional server hardware since applications will commonly have some units that benefit from tight-coupling of resources at the server scale. Future serverless computing abstractions might hide such details from programmers.

One challenge of disaggregation that may be instructive for serverless practitioners is the difficulty of microsecond-level latencies [30]. For small remote transfers, access latency can dominate completion time, potentially leading to stranded resources that cannot easily be scheduled for other tasks.

3. Research Synergies

The serverless computing perspective suggests concrete ways to inform disaggregated hardware research, and vice versa.

3.1. Compute Instance Launch

Startup times are a critical performance measure for cloud functions [31]. For example, they are especially important in machine learning serving when a framework provides inference on demand over a range of pre-trained models [32, 33]. Elasticity is provided by instantiating many replicas of a model and scaling their number in response to demand, so the time required to prepare replicas for serving determines tail latency behavior. Slow startup times may require over-provisioning of replicas and can make certain latency targets unattainable. With disaggregated hardware, pre-initialized models could reside in shared remote memory. Cache-like interfaces to remote memory (including paging or hardware-managed caches [34–38]) would enable near instant start times, followed by a brief performance penalty while the caches warm [39–42].

3.2. Communication and Ephemeral Data

Distributed big data processing frameworks [43, 44] often exchange large amounts of intermediate data between nodes of a computation graph. It has been common to materialize and store intermediates, both for fault tolerance and as a buffer to match upstream production and downstream consumption. Doing so can mean copying large data sets several times, first to a storage service, then to the consumer. The serverless data center is a natural fit for this scenario; disaggregated memory and storage allow intermediate data to be sent simply by passing a reference, an approach that benefits from the duality of storage and communication [45]. As an added benefit, senders can save their output and terminate, resuming only after other input required for further steps becomes available. Work in this area also includes Pocket [46], which focuses on scalability and fast metadata access for ephemeral data.

3.3. Persistent Data

The persistent storage options presently available to cloud functions have disappointing performance and cost characteristics, especially when compared to block storage available on VMs [1]. The desired solution would provide shared storage backed by multi-tenant resource pools, with caching, buffering, cost, and performance characteristics similar to those of local file systems (e.g., as proposed by LegoOS [7]). Here shared remote memory and a diversity of network-attached storage devices can provide a benefit. Research on high-performance tiered key-value storage [47] represents progress in this area.

3.4. Heterogenous Compute

Machine learning pipelines often require a mix of different compute platforms (e.g. CPUs, GPUs, TPUs). This accelerator landscape is evolving quickly, leading to deployment challenges. Disaggregated hardware allows flexible acquisition and allocation of compute resources, and the serverless programming model provides an abstraction that makes the necessary application decomposition natural. Google's TPU pods and Microsoft's Brainwave project demonstrate the power of network-attached accelerators [25, 48], and might work well in a serverless data center context.

3.5. Cloud Infrastructure

The multi-tenant nature of public clouds requires sophisticated management, which becomes subject to stringent performance demands and much more activity with serverless computing. Virtualization of cloud infrastructure has already benefited from hardware acceleration (e.g. the Amazon Nitro system [49]). Additional support will be needed to enforce fairness as more resources are shared. Similarly, security concerns permeate every part of the cloud and secure access to resources will need to be provided with low overhead. Encryption is one aspect, but so are authentication and key management (secure enclaves and trusted platform modules represent a start in this direction [50, 51]). Finally, cloud networks are heavily virtualized to provide secure and easy-to-use semantics to users [52]. Emerging networking technology may provide new options for enforcing these rules. For example, photonic circuit switching will create low-overhead paths between nodes in a virtual network and allow sub-microsecond reconfiguration times [53].

3.6. Coordination

Cloud functions create a problem that they are surprisingly ill-suited to resolve: they break up an application and induce logical disaggregation, but their limitations preclude running the distributed systems algorithms needed to coordinate among these pieces [28] (as described in Section 2, they are short-lived and do not accept inbound network connections). While other serverless computing models such as "cloud actors" might mitigate these problems, the increased need for coordination among distributed bits of applications may demand hardware solutions. These might build on prior research on network [54–56] and storage-based [57] coordination mechanisms.

3.7. Fault Tolerance

As described in Section 2, serverless applications can involve hundreds or thousands of independent nodes in a data center. This distributed setting makes fault tolerance, or at least fault visibility, critical. One approach is to allow applications to define fate-sharing rules which are then enforced by the network [58], which is an idea that serverless frameworks might incorporate. Many other opportunities exist in memory and storage disaggregation to provide flexible failure semantics to the user (e.g. transparent erasure coding).

4. Designing the Serverless Data Center

While many of the research directions proposed in Section 3 can be explored in isolation, we believe that hardware/software co-design has an important role to play in shaping future serverless data centers. Doing this requires the ability to make simultaneous changes across multiple layers of a system stack, possibly touching hardware, networks, operating systems, cloud services, and user applications, all in the context of realistic workloads. While prior work has shown that significant progress can be made using standard server hardware in a disaggregated fashion [7, 59], hardware simulation platforms will be needed to push designs further. FireSim and dist-gem5 can simulate distributed systems with a range of fidelity and simulation speed [60, 61]. FireSim, for example, uses FPGAs in the cloud to provide cycle-exact simulation of thousands of nodes, connected by a configurable network, at 100x-1000x slower than real time. Some serverless environments are available as open source [62–64], and can serve as research prototypes. Further development may be needed, e.g., to integrate state-of-the-art isolation techniques [65, 66] or improved scheduling algorithms. Combining serverless software frameworks with open simulation platforms will allow the serverless hardware and software communities to rapidly evaluate new networking technology, accelerators, or memory properties.

5. Conclusion

Serverless computing is emerging as an important paradigm that promises to shape the future of cloud computing. Driven by a desire to make cloud programming simpler, it provides abstractions that allow flexible, fine-grained, and elastic resource allocation. As a direct consequence of these aims, serverless applications are written in a logically diaggregated way and run in a physically distributed way, even in today's data centers; they are ready for disaggregated hardware. Serverless computing provides a viable evolutionary path for data center hardware, one that frees designers to develop the right hardware for the workload without being constrained by traditional server-based expectations. The future *serverless data center* will emerge from exploiting the synergy between hardware disaggregation and serverless computing.

Acknowledgements

We thank the anonymous reviewers and our colleagues for insightful feedback that helped shape this paper. This research was funded by NSF CISE Expeditions Award CCF-1730628, and by gifts from Alibaba, Amazon Web Services, Ant Financial, Arm, Capital One, Ericsson, Facebook, Google, Huawei, Intel, Microsoft, Nvidia, Scotiabank, Splunk, and VMware.

References

[1] 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.

[2] Krste Asanović. FireBox: A hardware building block for 2020 warehouse-scale computers. In *FAST 2014*, 2014.

[3] HP Labs. The Machine. https://www.labs.hpe.com/the-machine, 2017. Accessed: 2019-04-12.

[4] Michael P. Papazoglou and Dimitrios Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10):25–28, 2003.

[5] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Microservices architecture enables DevOps: Migration to a cloud-native architecture. *IEEE Software*, 33(3):42–52, 2016.

[6] Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.

[7] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying 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, 2018. USENIX Association.

[8] Zaid Al-Ali, Sepideh Goodarzy, Ethan Hunter, Sangtae Ha, Richard Han, Eric Keller, and Eric Rozner. Making serverless computing more serverless. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, page 456–459. IEEE, Jul 2018.

[9] Josep Sampé, Marc Sánchez-Artigas, Pedro García-López, and Gerard París. Data-driven serverless functions for object storage. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, Middleware '17, pages 121–133, New York, NY, USA, 2017. ACM.

[10] K. Katrinis, D. Syrivelis, D. Pnevmatikatos, G. Zervas, D. Theodoropoulos, I. Koutsopoulos, K. Hasharoni, D. Raho, C. Pinto, F. Espina, S. Lopez-Buedo, Q. Chen, M. Nemirovsky, D. Roca, H. Klos, and T. Berends. Rack-scale disaggregated cloud data centers: The dReDBox project vision. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 690–695, March 2016.

[11] Kevin Lim, Yoshio Turner, Jose Renato Santos, Alvin AuYoung, Jichuan Chang, Parthasarathy Ranganathan, and Thomas F. Wenisch. System-level implications of disaggregated memory. In *IEEE International Symposium on High-Performance Comp Architecture*, page 1–12. IEEE, 2012.

[12] Huawei. High throughput computing data center architecture - thinking of data center 3.0. https://www.huawei.com/ilink/en/download/HW_349607, 2014. Accessed: 2019-04-12.

[13] Intel. Intel rack scale design. https://www-ssl.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design-overview.html, 2017. Accessed: 2019-04-12.

[14] Facebook. Disaggregated rack. https://web.archive.org/web/20160421092139/http://www.opencompute.org/wp/wp-content/uploads/2013/01/OCP_Summit_IV_Disaggregation_Jason_Taylor.pdf, 2013. Accessed: 2019-04-12.

[15] Carsten Binnig, Andrew Crotty, Alex Galakatos, Tim Kraska, and Erfan Zamanian. The end of slow networks: It's time for a redesign. *Proc. VLDB Endow.*, 9(7):528–539, Mar 2016.

[16] Marcos K. Aguilera, Nadav Amit, Irina Calciu, Xavier Deguillard, Jayneel Gandhi, Pratap Subrahmanyam, Lalith Suresh, Kiran Tati, Rajesh Venkatasubramanian, and Michael Wei. Remote memory in the age of fast networks. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 121–127. ACM, 2017.

[17] Alex Netes. EDR Infiniband. In *11th Annual OpenFabrics Alliance Workshop*, 2015. Accessed: 2019-04-12.

[18] Chen Sun, Mark T. Wade, Yunsup Lee, Jason S. Orcutt, Luca Alloatti, Michael S. Georgas, Andrew S. Waterman, Jeffrey M. Shainline, Rimas R Avižienis, Sen Lin, Benjamin R. Moss, Rajesh Kumar, Fabio Pavanello, Amir H. Atabaki, Henry M. Cook, Albert J. Ou, Jonathan C. Leu, Yu-Hsin Chen, Krste Asanović, Rajeev J. Ram, Miloš A. Popović, and Vladimir M. Stojanović. Single-chip microprocessor that communicates directly using light. *Nature*, 528(7583):534–538, dec 2015.

[19] An Chen. A review of emerging non-volatile memory (NVM) technologies and applications. *Solid-State Electronics*, 125:25–38, 2016.

[20] Gen-Z Consortium. Gen-Z overview. Technical report, Gen-Z Consortium, 2016.

[21] Bruce Wile. Coherent accelerator processor interface (CAPI) for POWER8 systems. Technical report, IBM Systems and Technology Group, September 2014.

[22] Cache coherent interconnect for accelerators. https://www.ccixconsortium.com, 2017. Accessed: 2019-04-12.

[23] Kevin Lim, Jichuan Chang, Trevor Mudge, Parthasarathy Ranganathan, Steven K. Reinhardt, and Thomas F. Wenisch. Disaggregated Memory for Expansion and Sharing in Blade Servers. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 267–278, New York, NY, USA, 2009. ACM.

[24] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 7:1–7:13, New York, NY, USA, 2012. ACM.

[25] Eric Chung, Jeremy Fowers, Kalin Ovtcharov, , Adrian Caulfield, Todd Massengill, Ming Liu, Mahdi Ghandi, Daniel Lo, Steve Reinhardt, Shlomi Alkalay, Hari Angepat, Derek Chiou, Alessandro Forin, Doug Burger, Lisa Woods, Gabriel Weisz, Michael Haselman, and Dan Zhang. Serving DNNs in real time at datacenter scale with project Brainwave. *IEEE Micro*, 38:8–20, March 2018.

[26] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang. Applied machine learning at Facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629, Feb 2018.

[27] E. Frachtenberg. Holistic datacenter design in the Open Compute Project. *Computer*, 45(7):83–85, July 2012.

[28] Joseph M. Hellerstein, Jose Faleiro, Joseph E. Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. Serverless computing: One step forward, two steps back. In *CIDR*, 2019.

[29] A technical overview of the Oracle Exadata Database Machine and Exadata Storage Server. https://www.oracle.com/technetwork/database/exadata/exadata-technical-whitepaper-134575.pdf, 2012. Accessed: 2019-04-12.

[30] Luiz André Barroso, Mike Marty, David A. Patterson, and Parthasarathy Ranganathan. Attack of the killer microseconds. *Commun. ACM*, 60(4):48–54, 2017.

[31] Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. SOCK: Rapid task provisioning with serverless-optimized containers. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 57–70, 2018.

[32] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 613–627, 2017.

[33] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. TensorFlow-serving: Flexible, high-performance ML serving. *arXiv preprint arXiv:1712.06139*, 2017.

[34] 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, 2017. USENIX Association.

[35] Yongjun Lee, Jongwon Kim, Hakbeom Jang, Hyunggyun Yang, Jangwoo Kim, Jinkyu Jeong, and Jae W. Lee. A fully associative, tagless DRAM cache. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, page 211–222. ACM, 2015.

[36] Dejan S. Milojicic, David L. Black, and Steven J. Sears. Operating system support for concurrent remote task creation. In *IPPS*, 1995.

[37] Stavros Volos, Djordje Jevdjic, Babak Falsafi, and Boris Grot. An effective DRAM cache architecture for scale-out servers. Technical report, MSR-TR-2016-20, Microsoft Research, 2016.

[38] Evangelos P. Markatos and George Dramitinos. Implementation of a reliable remote memory pager. In *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ATEC '96, pages 15–15, Berkeley, CA, USA, 1996. USENIX Association.

[39] Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, and Felipe Huici. My VM is lighter (and safer) than your container. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 218–233. ACM, 2017.

[40] Horacio Andrés Lagar-Cavilla, Joseph Andrew Whitney, Adin Matthew Scannell, Philip Patchin, Stephen M. Rumble, Eyal De Lara, Michael Brudno, and Mahadev Satyanarayanan. SnowFlock: rapid virtual machine cloning for cloud computing. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 1–12. ACM, 2009.

[41] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[42] M. Satyanarayanan, Henry H. Mashburn, Puneet Kumar, David C. Steere, and James J. Kistler. Lightweight recoverable virtual memory. *ACM Trans. Comput. Syst.*, 12(1):33–57, February 1994.

[43] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[44] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[45] Michael Young, Avadis Tevanian, Richard Rashid, David Golub, Jeffrey Eppinger, Jonathan Chew, William Bolosky, David Black, and Robert Baron. The duality of memory and communication in the implementation of a multiprocessor operating system. In *In Proceedings of the 11th ACM Symposium on Operating Systems Principles*, pages 63–76, 1987.

[46] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. Pocket: Elastic ephemeral storage for serverless analytics. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 427–444, 2018.

[47] Chenggang Wu, Vikram Sreekanti, and Joseph M. Hellerstein. Autoscaling tiered cloud storage in Anna. *Proceedings of the VLDB Endowment*, 12, 2019.

[48] Zak Stone. Now you can train TensorFlow machine learning models faster and at lower cost on cloud TPU pods. https://cloud.google.com/blog/products/ai-machine-learning/now-you-can-train-ml-models-faster-and-lower-cost-cloud-tpu-pods. Accessed: 2019-04-12.

[49] Anthony Liguori. C5 instances and the evolution of Amazon EC2 virtualization. https://www.youtube.com/watch?v=LabltEXk0VQ, 2017. Accessed: 2019-04-12.

[50] Miriam Zimmerman. Virtual trusted platform module for shielded VMs: security in plaintext. https://cloud.google.com/blog/products/gcp/virtual-trusted-platform-module-for-shielded-vms-security-in-plaintext. Accessed: 2019-04-12.

[51] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptology ePrint Archive*, page 86, 2016.

[52] R. Jain and S. Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, November 2013.

[53] S. Moazeni, J. Henriksson, T. J. Seok, M. T. Wade, C. Sun, M. C. Wu, and V. Stojanović. Microsecond optical switching network of processor SoCs with optical i/o. In *Optical Fiber Communication Conference*, page Th1G.1. Optical Society of America, 2018.

[54] Zsolt István, David Sidler, Gustavo Alonso, and Marko Vukolic. Consensus in a box: Inexpensive coordination in hardware. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 425–438, 2016.

[55] Huynh Tu Dang, Daniele Sciascia, Marco Canini, Fernando Pedone, and Robert Soulé. NetPaxos: Consensus at network speed. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, page 5. ACM, 2015.

[56] Jialin Li, Ellis Michael, Naveen Kr. Sharma, Adriana Szekeres, and Dan R.K. Ports. Just say NO to Paxos overhead: Replacing consensus with network ordering. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 467–483, 2016.

[57] Eli Gafni and Leslie Lamport. Disk Paxos. In *International Symposium on Distributed Computing*, pages 330–344. Springer, 2000.

[58] Amanda Carbonari and Ivan Beschasnikh. Tolerating faults in disaggregated datacenters. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, pages 164–170, New York, NY, USA, 2017. ACM.

[59] Peter X. Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. Network Requirements for Resource Disaggregation. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 249–264, Savannah, GA, 2016. USENIX Association.

[60] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, Qijing Huang, Kyle Kovacs, Bora Nikolic, Randy Katz, Jonathan Bachrach, and Krste Asanovic. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 29–42, June 2018.

[61] A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, and N. S. Kim. dist-gem5: Distributed simulation of computer clusters. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 153–162, April 2017.

[62] Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Serverless computation with OpenLambda. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[63] Apache OpenWhisk. https://openwhisk.apache.org/. Accessed: 2019-04-12.

[64] Knative: Kubernetes-based platform to build, deploy, and manage modern serverless workloads. https://github.com/knative. Accessed: 2019-04-12.

[65] Open-sourcing gVisor, a sandboxed container runtime. https://cloud.google.com/blog/products/gcp/open-sourcing-gvisor-a-sandboxed-container-runtime. Accessed: 2019-04-12.

[66] Firecracker – lightweight virtualization for serverless computing. https://aws.amazon.com/blogs/aws/firecracker-lightweight-virtualization-for-serverless-computing/, 2018. Accessed: 2019-04-12.