What Can Chiplets Bring to Multi-Tenant Clouds?

Jiechen Zhao
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
jc.zhao@mail.utoronto.ca

Natalie Enright Jerger
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
enright@ece.utoronto.ca

Mingyu Gao
Institute for Interdisciplinary Information Sciences
Tsinghua University
Beijing, China
gaomy@tsinghua.edu.cn

***Abstract*—Recent trends in cloud hardware point towards the rise of specialization and disaggregation. Concurrently, adoption of multi-chiplet architectures is gaining momentum due to technology innovations. While we have witnessed cost-efficiency benefits of chiplet-based hardware architectures in commercial products, there have been few discussions around the impact of chiplets on the multi-tenant cloud model. Multi-tenancy promises transparent resource sharing, but also raises first-class concerns of security vulnerability and workload isolation. In our vision, now is the time to advocate that the evolution towards chiplet-based multi-tenant cloud could be essential and rewarding in the long run, for cloud-scale concerns such as tail latency, security, and resource management, besides cost-efficiency.**

***Index Terms*—Cloud Computing, Multi-Tenancy, Chiplets**

I. Introduction

How well cloud system designers can extract efficiency from cloud hardware largely determines cloud performance per unit cost. To maximize hardware resource utilization, the *multi-tenant cloud model* advocates to transparently share physical machines among a number of users. Multi-tenancy has successfully improved cost efficiency with higher resource utilization. However, such hardware sharing also mandates futuristic clouds to integrate architectures that elevate isolation and security as first-class design constraints.

Traditionally, the clouds deploy large-scale workloads using general-purpose commodity servers. Choosing off-the-shelf hardware well fits the requirements of workloads such as web search, which typically exhibit embarrassing parallelism and feature simple communication patterns [1]. In such cases, performance scaling has been successful with the number of servers scaled out, while cost growth has been acceptable due to Moore's Law. However, recently two key facts have motivated cloud hardware changes beyond these general-purpose servers. First, cloud software infrastructure researchers can no longer expect unlimited commodity hardware performance scaling in the post Moore's Law era. Second, a variety of workloads, e.g., big data and machine learning [2], [3], [4], [5], demand intensive and complicated computation and communication.

Simply put, cloud hardware architectures are experiencing two new trends: specialization [6], [7] and disaggregation [8], [9]. Specialization optimizes a performance-dominant software/system component into a specialized silicon module. Disaggregation separates compute, memory, and storage into pools connected by fast networks and operates on big data. New architectural innovations are required that can (1) continue integrating more functionalities beyond the reticle limits of die sizes, (2) accommodate more heterogeneous IPs under low integration complexity, (3) enable greater memory integration and more efficient communication in big data era, (4) combat skyrocketing manufacturing costs in more economical ways.

Fortunately, the change from monolithic chip architectures to *chiplets* has the promise to fundamentally alter future cloud hardware. Chiplet-based systems feature small dies and chip resource disaggregation (details in Sec. II-B), and have successfully pushed the change from *silicon-centric thinking* to *system-level planning* and *IC-package co-design*. Such a design paradigm is naturally beneficial for saving manufacturing and non-recurring engineering (NRE) cost, processing data near the memory, customizing communication support, and facilitating heterogeneous integration. Thus, performance per dollar can continuously increase.

Interestingly, although chiplets have been extensively discussed among system architects, hardware designers, and SoC vendors, **there have been few discussions about the impact of chiplets on the multi-tenant cloud model**. Specifically, the key characteristic of chiplet-based architectures — physical isolation of resources previously belonging to a monolithic die — has the potential to offer better workload isolation and security mechanisms desired by the multi-tenant clouds. This paper thus tries to establish the synergy between chip disaggregation and multi-tenancy to benefit futuristic hyperscale cloud services. In our vision, it is worthwhile to explore chiplet-cloud co-design to investigate the benefits around (but not limited to) metrics that the cloud cares about, e.g., tail latency, security mechanisms, DRAM bandwidth scaling and isolation, as well as flexible communication support.

II. Background and Motivation

A. Trends of Cloud Hardware

Cloud hardware design becomes important when we are not considering *only* using commodity servers to serve users. There are currently several new hardware trends in the cloud. First, cloud architectures are adopting, and in some cases pioneering, a variety of devices, such as GPUs, FPGAs,

SmartSSDs, and SmartNICs. Second, introducing new hardware is usually motivated by specializing a performance-dominant component onto a budget of transistors. For example, cloud hardware has spurred new capabilities with specialized chips for machine learning, training, and inference [6], [7]. In addition, cloud system designers tend to offload key software components, e.g., in the OS or the network protocol, to hardware [10], [11], [12]. Third, different types of resources, including compute, memory, and storage, are disaggregated into fabric-attached pools that each can be scaled out independently per workload needs. Note that chip disaggregation discussed next would add extra cost-efficiency benefits even for newly adopted specialized architectures and disaggregated resources, especially the high-end products that require smaller technology nodes [13].

B. Chip Disaggregation + Cloud: Why Now?

Disaggregating high-end large-die servers to smaller chiplets is a boon to an increasing number of cloud hardware products, including general- and special-purpose processors. The state-of-the-art 96-core EPYC Genoa [14] contains twelve compute dies and one I/O die on an interposer. Recently, based on Foveros [15] and EMIB packaging technologies, Intel launched Sapphire Rapids Xeon CPUs [16] based on chiplets and coupled with four high-bandwidth memory (HBM) stacks. Meanwhile, Marvell is working on chiplet-based 50Tbps switches that provide more chip edges for SERDES I/Os [17] to overcome the reticle limit. Accelerators, such as FPGAs [18], ASICs [13], and GPUs [19] also have found their way into commercial chiplet-based architectures.

In a nutshell, the chiplet design philosophy is being increasingly leveraged in more and more commerial products due to the following two key characteristics:

1) **Smaller dies** have lower manufacturing cost due to fewer defects per die and more dies per circular wafer. Both improve the yield. In [20], die size of 300mm^2 yields 75.7% good chips, while 250mm^2 yields 94.2%.
2) **Chip resource disaggregation** reduces both manufacturing and design cost, and enables a more flexible SoC ecosystem. Chiplets are easier to validate due to fewer functions and smaller scales, and become increasingly modular for a broad range of SoCs. Thus, chiplet designs can be reused more easily, which leads to validation, verification, and IP cost savings, in addition to cheaper mask cost when using older technology nodes. Also, vendors do not waste silicon but can sell everything they fabricate by binning high-end and low-end chiplets, instead of downgrading an entire large die that potentially has many defects. Ultimately, developments and upgrades of different chiplets can be in their own cycle. Such an ecosystem encourages third parties to collaboratively customize SoC solutions tailored to different domains and market segments.

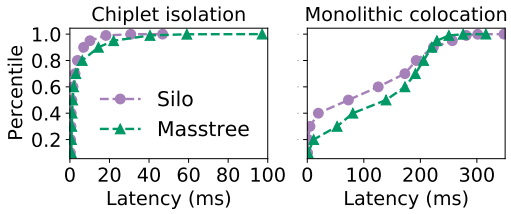

Fig. 1. CDF of latency comparison between chiplet-based and monolithic architectures. The policies of workload mapping are (a) one workload per 20-core chiplet with 14MB LLC, (b) colocating two workload on a monolithic server (40-core, 28MB).

III. Benefits of Chiplets in Clouds

A. Shorter Tails for Latency-Sensitive Workloads

Typically one physical machine in the cloud would host a number of co-located VMs to increase resource utilization. However, latency-sensitive applications mandate fine-grained resource isolation , otherwise interference between VMs would hurt their tail latencies [21]. In addition, virtualized environments spend extra system resources running hypervisors, which exacerbates the unexpected and hard-to-manage queuing effects. In practice, cloud providers sacrifice resource efficiency to handle one latency-critical workload per host to offer satisfactory tail latency guarantees [21].

We advocate that chiplet-based architectures have the potential to naturally provide physical workload isolation among some shared resources. Fig. 1 demonstrates the potential of disaggregation as a promising approach to removing interference on existing tightly-coupled CPU resources. We co-locate Silo [22] and Masstree [23], two in-memory latency-sensitive services, in both a monolithic host and an emulated host with the same amount of resources split across two identical chiplets. The figure shows that chiplet isolation can improve 99% latency of Silo/Masstree by 15/6×, mainly resulting from less last-level cache (LLC) and memory contention and consequently less queuing effects.

Although isolation can also be achieved by OS- or hardware-based low-level primitives, chiplet-aware scheduling is a complementary approach worth the attention. First, chiplet-based application scheduling can relieve the complexity of low-level resource management. For example, our experiment results of the chiplet isolation setting shown in Fig. 1 do not rely on any OS- or hardware-based resource partition. Instead, we simply perform a chiplet-application mapping and deliver similar to performance when workloads running standalone. The second lesson we learn is that blindly scaling one type of resource (LLC in this case) does not help if not removing interference from other workloads. In our experiment, the cache miss rates of Silo and Masstree are high, which indicates that LLC may be a significant bottleneck in the monolithic colocation setting. However, it turns out that chiplet-based isolation behaves much better than the monolithic setting, even though it only has half-sized LLC per workload.

B. Security Augmentation

Fine-grained resource sharing can leak sensitive information such as confidential data and service placement over side channels and/or covert channels. Existing techniques are insufficient. Common architectural solutions attempt to address some of the security concerns through thread pinning, memory bandwidth isolation, and network and cache partitioning. But they also inevitably sacrifice utilization and cost-efficiency [24].

Chiplet-based architectures have the opportunity to improve security because each chiplet has cores, caches, and memory interfaces physically isolated from the others. Such a natural physical isolation precludes many side-channel attacks that leverage shared microarchitectural states. With techniques such as restricted coherence domains [25] and simple chiplet-aware deployment policies, the resources of one tenant can be better protected from other tenants, without requiring conventional isolation techniques with degraded performance and cost-efficiency.

C. Memories Closer to Heterogeneous Compute

Chiplet-based packaging enables compute devices (e.g., SmartNICs and GPUs) to sit closer to high-bandwidth stacked HBMs[1] and enables shorter and denser signal routing between the compute and memories than traditional chip packaging. First, stacked HBMs are physically "closer" to devices than conventional DRAM buses, therefore dramatically alleviating the bandwidth limitation of the device, and reducing non-coherent PCIe transactions to the host memory[2]. Second, coherence can be more easily achieved among devices and the host [27], making the data movement entities "closer" to each other from application developers' perspective. Even a unified memory interconnect among CPUs and devices can be realized [12]. Furthermore, in multi-tenant settings, this architecture can reduce memory contention due to more chip edges and more isolated interfaces.

We created a microbenchmark that reads/writes randomly generated physical addresses on a remote server's memory. The read/write ratio is set to be 50%/50% and the payloads are set to be 512B. Each memory access request arrives at the remote server under Poisson process that represents typical cloud request statistical distribution at end-host servers [28]. We compare our setting that supposes infinite memory bandwidth for accessing in-memory data to a baseline remote server setting that resembles commodity servers. For the memory system of our baseline setting, we simulate the NIC to be similar as a Mellanox BlueField SmartNIC [29], which contains one on-board DDR4 channel and 16GB memory. We further optimize the baseline by enabling the NIC to be able to cache recently-accessed payloads in its on-board memory. This optimization saves the latency of fetching data from the host memory through a PCIe 3.0 x8 bus. We assume PCIe

[1]Each HBM2 stack can achieve 256–538 GB/s [26].

[2]For cache line size packets, DMA performance is bound by the μs-scale PCIe latency and DMA engine parallelism.

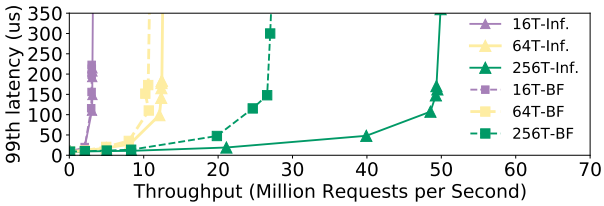

Fig. 2. Memory bandwidth limitations of (infinite vs. Mellanox BlueField (BF) configuration [29], T: thread).

can reach its theoretical latency and bandwidth, and faithfully simulate memory contention at both channel- and bank-level.

Fig. 2 illustrates that memory bandwidth becomes the bottleneck at high request rates, which limits system throughput@SLO[3] by more than 2× when compared with our baseline setting. Therefore, it is vital to exploit a more decoupled architecture like chiplet-based architecture to move memories closer to the compute for sufficient memory bandwidth and minimal memory contention.

D. Flexible and High-Speed Chip-to-Chip Interconnects

Designers can explore additional flexibility to customize inter-chiplet interconnects. There is still no standardized die-to-die communication protocol, but lately, open-source coherent interconnect standards that allow off-chip accelerators to interface the host are rapidly evolving, such as CXL [27] and OpenCAPI [30]. Additional interconnects such as Gen-Z [31] can connect servers through Ethernet fabrics. For examples, an OpenCAPI FPGA stack instance interfacing an IBM POWER9 processor can achieve up to 200Gbit/s bandwidth [9]. We show a few inter-chiplet interconnects with different performance and scalability in Table I.

The on-substrate interconnect provides additional routing capability [20]. It can efficiently accommodate the growing communication demands due to chip disaggregation, isolated away from the intra-chiplet network-on-chip (NoC). Moreover, as communication patterns become irregular and intensive, chiplet-based architectures can localize short-distance data movement within each chiplet, and use available inter-chiplet interfaces to customize interconnects for different workloads. For instance, we customized a high-bandwidth inter-chiplet interconnect, ButterDonut [32], that outperforms coherent Mesh on a microbenchmark. The number of chiplets are assumed to be 16, where each chiplet contains 16 CPU cores, and the microbenchmark settings are the same as that in Sec. III-C. The microbenchmark is ported into BookSim [33] to evaluate the corresponding interconnect performance in different interconnect designs. Compared with a 4×4 coherent Mesh, ButterDonut improves request throughput by 1.14×. Moreover, we test that the interconnect throughput is bottlenecked by the ingress/egress traffic between the network interface I/O chiplet and compute chiplets. Therefore, we isolate the traffic from the microbenchmark-induced memory traffic

[3]Service level objective (SLO) is set to be 30μs for 99th percentile latency [28].

TABLE I
INTER-CHIPLET INTERCONNECTS COMPARISON. ASSUME PCIE 4.0X16, NVLINK 50GB/S.

Interconnects	Protocol	Link/PHY	B/W	Multi-hop Latency	Deployment Scale
OpenCAPI	P2P	PCIe/ NVLink	32-50GB/s	Low/Medium	Package-level
CXL		PCIe/CXL	32GB/s		
ButterDonut	Packet-switched	Interposer	12x16GB/s (bisection)	Low	
Gen-Z		Ethernet Fabric	100Gbps+	Medium/High	Rack-level

by providing separate interconnect links. Our customization achieves 1.44× throughput improvement over the Mesh due to less interconnect contention. In our vision, there would be a diversity of interconnect design opportunities to satisfy domain-specific communication patterns.

IV. CONCLUSIONS

Multi-tenant clouds and chip disaggregation are recent movements that have progressed largely independently, even though they share a common goal: to extract the maximum transistor efficiency with the minimum cost. This paper summarizes a list of reasons to co-design chiplet-based cloud hardware with multi-tenancy. While this may take time to emerge, there are immediate opportunities for ideas from chip disaggregation to inform the progression of more cost-efficient and easy-to-manage multi-tenant clouds.

There are three takeaways.

- First, this idea showcases the new design philosophy for future cloud systems: hardware architects cannot be unaware of hyperscale software requirements, and system researchers can no longer follow the traditional approach of building each layer of the system stack separately. Instead, we must rethink the synergy between the software and hardware worlds from the ground up.
- Second, the similar purpose of multi-tenancy and chip disaggregation motivates co-design efforts, because multi-tenant clouds deploy layers of software that *logically partition* warehouse-scale computers into isolated resource groups for elastic usage among tenants, whereas chip disaggregation also "coincidentally" *physically partitions* tightly-coupled resources on a large server die into multiple lower-cost chiplets, but exposing this architecture to system software still as a virtually monolithic machine.
- Third, chiplet-aware cloud system design has the potential to benefit tail latency, security, memory and interconnect designs, and also chiplets' own original benefits — improved performance per dollar.

REFERENCES

[1] Luiz André Barroso, Jeffrey Dean, and Urs Holzle. Web search for a planet: The google cluster architecture. *IEEE micro*, 23(2):22–28, 2003.

[2] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[3] Mingyu Gao, Grant Ayers, and Christos Kozyrakis. Practical near-data processing for in-memory analytics frameworks. In *2015 International Conference on Parallel Architecture and Compilation (PACT)*, pages 113–124. IEEE, 2015.

[4] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 561–577, 2018.

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[6] Using tpus to train your model. https://cloud.google.com/ai-platform/training/docs/using-tpus#:~:text=Tensor%20Processing%20Units%20%28TPUs%29%20are%20Google%27s%20custom-developed%20ASICs,you%20don%27t%20need%20to%20manage%20the%20TPU%20yourself.

[7] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, et al. A configurable cloud-scale dnn processor for real-time ai. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–14. IEEE, 2018.

[8] 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, 2016.

[9] Christian Pinto, Dimitris Syrivelis, Michele Gazzetti, Panos Koutsovasilis, Andrea Reale, Kostas Katrinis, and H Peter Hofstee. Thymesisflow: A software-defined, hw/sw co-designed interconnect stack for rack-scale memory disaggregation. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 868–880. IEEE, 2020.

[10] Adrian M Caulfield, Eric S Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, et al. A cloud-scale acceleration architecture. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

[11] Irina Calciu, Ivan Puddu, Aasheesh Kolli, Andreas Nowatzyk, Jayneel Gandhi, Onur Mutlu, and Pratap Subrahmanyam. Project pberry: Fpga acceleration for remote memory. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 127–135, 2019.

[12] Nikita Lazarev, Neil Adit, Shaojie Xiang, Zhiru Zhang, and Christina Delimitrou. Dagger: Towards efficient rpcs in cloud microservices with near-memory reconfigurable nics. *arXiv preprint arXiv:2007.08622*, 2020.

[13] Yakun Sophia Shao, Jason Cemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, et al. Simba: scaling deep-learning inference with chiplet-based architecture. *Communications of the ACM*, 64(6):107–116, 2021.

[14] Amd® epyc™ genoa. https://en.wikichip.org/wiki/amd/cores/genoa.

[15] Intel® foveros. https://newsroom.intel.com/wp-content/uploads/sites/11/2019/08/Intel-Lakefield-HotChips-presentation.pdf.

[16] Intel® xeon™ sapphire rapids. https://www.intel.com/content/www/us/en/newsroom/news/new-intel-xpu-innovations-target-hpc-ai.html#gs.d8v4xu.

[17] Marvell® chiplet-based network switch. https://events.meptec.org/wp-content/uploads/Road/2021/RoadtoChiplets2021Sandeep_Bharathi.pdf.

[18] Intel® agilex™ fpga. https://www.intel.com/content/www/us/en/products/details/fpga/agilex.html.

[19] Patent of chiplet-based amd gpu architecture. https://patents.google.com/patent/WO2021016273A1/en.

[20] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H Loh. Enabling interposer-based disintegration of multi-core processors. In *2015*

48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 546–558. IEEE, 2015.

[21] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving resource efficiency at scale. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 450–462, 2015.

[22] Stephen Tu, Wenting Zheng, Eddie Kohler, Barbara Liskov, and Samuel Madden. Speedy transactions in multicore in-memory databases. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 18–32, 2013.

[23] Yandong Mao, Eddie Kohler, and Robert Tappan Morris. Cache craftiness for fast multicore key-value storage. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 183–196, 2012.

[24] Christina Delimitrou and Christos Kozyrakis. Bolt: I know what you did last summer... in the cloud. *ACM SIGARCH Computer Architecture News*, 45(1):599–613, 2017.

[25] Yaosheng Fu, Tri M Nguyen, and David Wentzlaff. Coherence domain restriction on large scale systems. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 686–698. IEEE, 2015.

[26] Samsung first 3rd-generation (16gb) hbm2e. https://news.samsung.com/global/samsung-to-advance-high-performance-computing-systems-with-launch-of-industrys-first-3rd-generation-16gb-hbm2e.

[27] Cxl consortium. compute express link (cxl). https://www.computeexpresslink.org/.

[28] George Prekas, Marios Kogias, and Edouard Bugnion. Zygos: Achieving low tail latency for microsecond-scale networked tasks. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 325–341, 2017.

[29] Nvidia® mellanox® bluefield® smartnic for ethernet. https://www.nvidia.com/en-us/networking/products/data-processing-unit/.

[30] Opencapi. open coherent accelerator processor interface (opencapi). https://opencapi.org/.

[31] Gen-z consortium: Computer industry alliance revolutionizing data access. http://genzconsortium.org/.

[32] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H Loh. Exploiting interposer technologies to disintegrate and reintegrate multicore processors. *IEEE Micro*, 36(3):84–93, 2016.

[33] Nan Jiang, Daniel U Becker, George Michelogiannakis, James Balfour, Brian Towles, David E Shaw, John Kim, and William J Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *2013 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pages 86–96. IEEE, 2013.