Server Architecture from Enterprise to Post-Moore

Babak Falsafi[1], Michael Ferdman[2], Boris Grot[3]
[1]EPFL [2]Stony Brook University [3]University of Edinburgh

***Abstract*—Luiz Barroso started his career at DEC, investigating workload-optimized multiprocessor server architectures marketed to enterprises in the 1990s. These high-margin low-volume products lost their market to more cost-effective enterprise servers built from high-volume desktop CPUs riding Moore's Law. The enterprise market has slowly transitioned to the cloud, where desktop PCs have formed the backbone of computing in datacenters since the early 2000s to minimize cost and maximize the return on investment. Moving forward, with the absence of Moore's Law, future servers require a clean-slate cross-stack design to scale in compute, communication and storage capacity while reducing operational, capital and environmental costs.**

***Index Terms*—datacenters, server architecture, workloads, metrics**

I. Introduction

Today, datacenters form the backbone of all digital services. Datacenter growth is fueled by the global availability of online services, digitalization of industries, sustained data deluge, and Artificial Intelligence. At the heart of datacenters are volume servers that trace their roots back to the desktop PCs of the 90s, running Linux and other open source software. Scaling computing with inexpensive PCs as building blocks to maximize the return on investment is analogous to RAID storage in the 1980s replacing high-margin enterprise disks with an array of cost-effective PC disks.

For the past two decades, datacenters have enjoyed riding Moore's Law and improvements in chip density and efficiency to scale their services while minimizing construction growth and overall energy consumption. Modern datacenters heavily rely on consolidation of workloads, virtualization, compression, and emerging software construction paradigms (e.g., microservices, serverless) to maximize server utilization while maintaining service-level contracts.

Unfortunately, with a slowdown in Moore's Law[1] and chip efficiency, power consumption in servers is witnessing an unprecedented growth. Figure 1 depicts Thermal Design Power (TDP), a measure for the maximum amount of heat a chip can dissipate, for several generations of CPUs and GPUs from 2008 through 2023. While the figure does not plot TDP ratings before 2008, CPU TDPs remained flat, at 100-150W, for roughly two decades. Since 2019, CPUs are experiencing a sharp rise in TDP suggesting that future improvements in design will be accompanied with higher power consumption with implications on datacenter growth. GPUs' TDP is rising even faster because of their much higher arithmetic density as compared to CPUs.

[1]$https://en.wikipedia.org/wiki/Transistor\_count.$

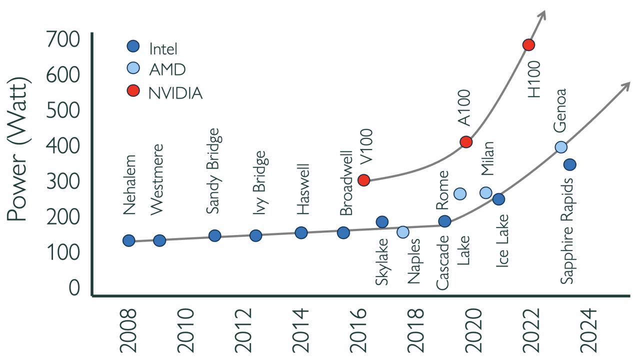

Fig. 1. Increasing TDP in server CPUs and GPUs.

The insatiable appetite for computing continues to grow, driven by, on one hand, the expansion in data collection, storage, and analysis, and on the other, the proliferation of artificial intelligence. In the absence of Moore's Law, these trends demand building and operating more servers, with a commensurate impact on overall server energy consumption and environmental impact through operational and embodied emissions.

To maximize silicon efficiency, post-Moore servers require a clean-slate approach to the design of the computing stack, from algorithms down to hardware. Much like the enterprise multiprocessor servers of the 1990s, post-Moore server design requires proper resource provisioning for workloads, and precise metrics and measurement methodologies to quantify a server design's output based on the provisioned resources.

In this article, we reflect on the convergence of server architecture from integrated enterprise-class multiprocessors of the 1990s to the modern-day cloud server architecture. We then present requirements for metrics to evaluate the performance and efficiency of post-Moore servers, emerging workloads, services and computational paradigms in datacenters, and trends in the design of post-Moore servers.

II. From Enterprise to Cloud

Server architecture has undergone a major transformation from the enterprise servers of the 90s to the cloud servers today. These transformations have been the result of exponential improvements in CPU performance and memory density, thanks to Moore's Law, the demand for scalability from enterprise IT to public cloud, and the focus on return on investment in datacenters. In this section, we will briefly go over these transformations, Luiz's contributions to these transformations, and our work during this period on introducing cloud-native server architecture.

A. Enterprise Servers

The primary driver for server architecture innovation in the 1990s was commercial server software, initially in the database management market and eventually the web server market. With increasing data volumes, the growing gap between DRAM and disk, and exponential reductions in the cost of DRAM, these workloads primarily became in-memory.

At the higher end of the premium enterprise servers, vendors like Sun Microsystems, HP, DEC, and Silicon Graphics started building mid- (e.g., 4-16 CPUs) to large-scale (e.g., 128 CPUs) shared-memory multiprocessors to enhance memory capacity and ride Moore's Law with DRAM and RISC CPUs. At the lower end of the premium market, x86 servers, whose CPU performance was competitive with their RISC counterparts, was riding the same Moore's Law while benefiting from the economies of scale of desktop CPU volumes.

Many in the academic community were evaluating commercial server workloads on enterprise servers of the time and pointing out the mismatch between the workloads' requirements and the CPU microarchitecture, leading to underutilized silicon. These studies indicated that commercial server workloads benefited from more capacity in the cache hierarchies, because of larger instruction and data working sets, but showed limited instruction-level parallelism due to dependent accesses to memory.

Luiz was among those who carefully evaluated this mismatch [2], and then argued and demonstrated through DEC Piranha the case for custom multicore processors for commercial workloads. These multicore processors could exploit thread-level (rather than instruction-level) parallelism with older generation (scalar) RISC pipelines and unusually large cache hierarchies required for commercial workloads.

While RISC CPUs had a phenomenal penetration in the embedded and mobile platform market, they eventually lost ground to x86 CPUs in the enterprise server market. The RISC servers' high margins and low volumes rendered them economically unsustainable. Instead, Intel and AMD created a portfolio of CPUs that included enterprise server CPUs to help transition the market to x86.

B. Transition to Cloud

The decade of 2000 witnessed the emergence of datacenters as warehouse-scale computers, built with stripped-down volume servers interconnected with commodity network gear to implement massive internet services such as search, online retail, and social networks. The key goal was to build services with the cheapest building blocks available at the time (i.e., the desktop PCs) to maximize return on investment. The servers ran Linux and open-source software and relied on distributed computing to exploit parallelism and provide fault tolerance.

A key mismatch between volume servers and the services they were running was rooted in the tight latency requirements in service-level agreements with customers [4]. The desktop PC and its OS were either running in the nanosecond time-scale computing and accessing memory, or in the millisecond time-scale accessing a hard disk through the OS [1]. Meanwhile, the data that services were hosting in memory was made available through the networks, where switch-based fabrics operated in the microsecond time-scale. Therefore, all OS overheads (e.g., thread context switches, interrupts, and network communication software stacks) in the microsecond time-scale were directly on the critical path of the services.

IT companies scaled their services throughout this decade by building more and larger datacenters at an unprecedented pace while maintaining low server utilization to guarantee service-level agreements (SLAs). Volume servers are inherently not energy-proportional and, as such, low utilization implies high electricity consumption. More datacenters also meant higher capital expenditure and overall lower return on investment.

In the decade of 2010, datacenter operators continued riding Moore's Law with DRAM and CPUs, focusing on efficiency in infrastructure to maximize electricity delivery and optimizing software stacks to increase performance per dollar while maintaining SLAs. The two key technologies that improved server utilization were consolidation of workloads and virtualization, allowing for multi-tenancy to sell unused resources to cloud customers.

C. Cloud-Native Server Architecture

With Dennard Scaling ending and Moore's Law slowing down, many academics continued evaluating opportunities to design servers that matched commercial server software requirements, including database and web workloads. These included analyses of the core complexity, various forms of multi-threading, and capacity provisioning in multicore cache hierarchies.

In 2011, Hardavellas et al. [8] revisited workload-centric server architecture. They demonstrated that, for the combination of stringent chip power constraints, emerging high-bandwidth and energy-efficient memory fabrics, and the abundance of request-level parallelism in server workloads, a custom manycore CPU would be optimal for throughput, power, and area in servers. Unlike conventional CPU designs, such custom CPUs would have minimal on-chip memory (i.e., MBs) to hold the instruction working set of deep server software stacks, and minimal complexity (i.e., power, area) cores to access off-chip data and exploit request-level parallelism across server threads.

Ferdman et al., [5] studied the mismatch between volume server architecture in the cloud and scale-out services based on open-source software stacks. In particular, they listed several microarchitectural characteristics salient in scale-out workloads that were drastically different from desktop workloads: (1) instruction supply bottleneck due to large instruction working sets in server software stacks, (2) low instruction-level (ILP) and memory-level (MLP) parallelism in server software stacks, (3) secondary data working sets that are orders of magnitude larger than on-chip memory, and (4) low on- and off-

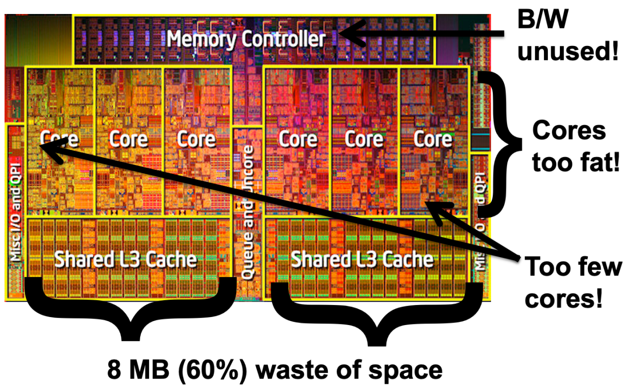

Fig. 2. Server CPU microarchitecure/workload mismatch.

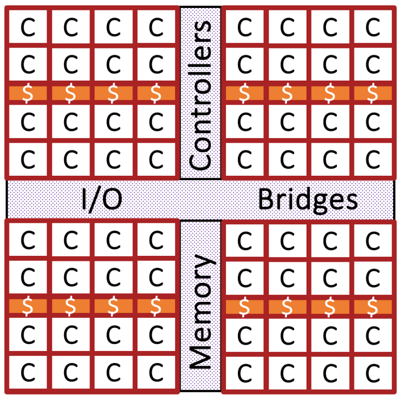

Fig. 3. A scale-out processor with 4 16-core pods.

chip per-thread memory traffic. The workloads were also found to make minimal use of floating-point arithmetic and vector operations. These characteristics were at odds with state-of-the-art volume servers at the time, which were being shipped with half a dozen ILP-centric desktop x86 cores padded with 12MB of LLC per socket, with two sockets serving 10s of GB of DRAM. Figure 2 illustrates the silicon mismatch between the needs of scale-out services and a Westmere die.

These results laid the foundation for the first-generation of cloud-native server CPUs [9] called *Scale-Out Processors*. Scale-out processors integrated multiple server "pods" sharing memory and I/O ports on a single die. Figure 3 illustrates the anatomy of a 64-core scale-out processor with four pods. Each pod was optimized for throughput per area of silicon and uses 3-way out-of-order 64-bit ARM cores modeled after Cortex A15, the flagship out-of-order ARM core at the time. The L2 was judiciously sized to capture the instruction working set and accelerate instruction supply while accommodating data reuse in the OS and application metadata. The cores were interconnected with a crossbar to statically interleaved L2 banks, memory controllers and I/O bridges.

Lotfi-Kamran et al. further studied the on-chip network traffic for scale-out workloads and concluded that there is little (e.g., $< 2\%$ on average) on-chip coherence traffic. The majority of traffic is either supplying instructions from the LLC, or checking whether data is available on-chip and fetching it from off-chip, because of the large disparity between on-chip and off-chip memory capacity. To maximize silicon efficiency and optimize interconnect topology, the authors proposed a custom on-chip interconnect for each pod to connect the cores in a column of request/reply relays to statically interleaved L2 banks. The L2 banks were fully connected with memory controllers and I/O bridges through a flattened butterfly network.

Each pod ran a full software stack and had its data sharded across physical memory partitions. Because pods operated as independent servers with no contention on microarchitectural resources, optimally sizing a pod and scaling the number of pods per chip would allow for a linear increase in throughput while maintaining the proper core-to-cache ratio and optimizing overall datacenter costs at the board and system level.

Compared to the contemporary designs, Scale-Out Processors provided a much higher level of thread-level parallelism, more silicon area given to cores than caches, and a much faster instruction supply. The first-generation of Cavium ThunderX followed this chip organization with a crossbar, in-order MIPS cores (retrofitted with an ARM decoder) and larger L1 instruction caches for better instruction supply. Fast forward to now, with Intel announcing their Sierra Forest line of cloud-optimized processors featuring 144 single-threaded cores organized in 4-core clusters that share an L2, and a shared LLC with a per-core capacity under 1MB. The architecture of Sierra Forrest reflects the design philosophy of Scale-Out Processors in terms of more cores, leaner cores, and less cache per core compared to conventional server CPUs.

III. Metrics

Regardless of the goal of a processor design the primary target is always to maximize performance, while staying within the target engineering costs and technology limitations. As such, a principled approach to designing next-generation cloud-native processors must first define and measure performance, and then incorporate performance into metrics that will be used to optimize performance within the constraints of capital and operational cloud server expenditure.

A. Defining Performance

Luiz's work at DEC was among the first to consider servers as a separate class of systems with a unique set of characteristics [2], demonstrating that server performance is dominated by memory stalls and limited instruction level parallelism. On real systems, a barrage of simulated client requests can be used to assess the peak throughput the server can achieve measured in queries or requests served per second. In simulation, measuring multiple server requests end-to-end would be prohibitively slow. Moreover, the de facto CPU performance metric, IPC (instructions per cycle) allowing for fine-grain measurements of tens to hundreds of thousands of instructions at a time, would not be applicable in multiprocessor server workloads because synchronization in the software stack with threads spinning perturbs IPC.

While metrics such as MLP (memory level parallelism) could help indicate memory bottlenecks in workloads, they fell short of offering tight estimates on overall performance. Wenisch et al., [11] instead showed that *U-IPC*, the number of user instructions per cycle, is proportional to server throughput for a wide range of server applications, such as running OLTP, DSS and web workloads generating dynamic content, because threads spin in the OS. As such, U-IPC could be used reliably as a microarchitectural metric for throughput with simulation.

With the transition to the cloud and the birth of global IT services from search, to social media, retail and media, customer experience in terms of response time became a negotiated metric in cloud Service Level Agreements (SLAs). Strict quality of service (QoS) requirements were set on response time of servers, setting bounds not just on the average latency, but on request tail latency. Queuing theory dictates that tail effects become prominent at high server utilization. As a direct consequence, cloud operators must sacrifice throughput to meet tail latency constraints, intentionally running servers at reduced utilization to meet latency targets.

Today, the key metrics for cloud workloads are peak throughput at a given QoS target, where the target is defined as a maximum acceptable end-to-end latency for a specified fraction (e.g., 95th percentile) of the requests. Luiz's work at Google pointed to the cumulative impacts of tail latencies at scale compounding into large end-to-end latencies. His work showed the critical impact of tail latencies in datacenters [4], where backend services must meet much stricter QoS targets (e.g., 99.9th percentile) to permit end-to-end latency targets (e.g., 95th percentile) to be met. In practice, production servers rarely operate at their peak theoretical throughput, highlighting the need to design processors that specifically target the relevant metrics so as to optimize for efficiency and overall cost.

B. Silicon Efficiency

The widening gap between mainstream CPUs and the needs of cloud workloads inspired a critical look at how to quantify the performance and efficiency of cloud-native processors. With silicon area being a direct reflection of the processor cost, we proposed the *performance density* metric, which relates the QoS-conscious peak throughput that can be achieved by a square millimeter of silicon. Using performance density as a roadmap to understanding the cloud-native CPU landscape suggests processors with many lean cores and limited cache capacity [9]. The reluctance of the dominant server CPU vendors to shift toward designs with higher performance density and make more effective use of silicon led to a rapid emergence of a number of competing CPU designs targeting the cloud market, including the Cavium ThunderX, Amazon Graviton, link Altra, and Huawei TaiShan.

As chip power envelopes continue their exponential increase and Moore's Law crawls toward its ultimate demise, cloud-native processor design is approaching another inflection point.

	Throughput/mm^2		**Throughput/Watt**	
	Zen 3	**Altra**	**Zen 3**	**Altra**
Data Analytics	1.38	3.87	1.46	2.53
Data Caching	1.28	2.06	1.35	1.34
Data Serving	1.22	1.94	1.29	1.27
DSB Media Service	0.89	1.53	0.95	1.00
Graph Analytics	1.55	3.31	1.65	2.16
In-memory Analytics	1.52	3.86	1.61	2.52
Media Streaming	1.09	1.82	1.16	1.19
Web Search	1.46	4.06	1.54	2.65
Web Serving	1.37	4.04	1.46	2.64

TABLE I
SINGLE-CORE PERFORMANCE DENSITY AND PERFORMANCE/POWER FOR AMD ZEN 3, AMPERE ALTRA NORMALIZED TO INTEL ICE LAKE.

Integrating more silicon per server node increases fabrication cost, but with increasing clock frequencies and diminishing improvements in memory density, the design constraints shift away from processor silicon area, toward overall power consumption and memory bandwidth.

Table I compares performance density and performance/power on CloudSuite 4.0 and DeathStarBench Media Service for a single core of AMD Zen 3 and Ampere Altra, both normalized to Intel Ice Lake. Ice Lake, Zen 3, and Altra are 6.2, 4.0 and 1.40 mm^2 in area per core in comparable technology nodes. The TDP of each chip is divided by the number of cores to get 5.78, 3.52, and 2 watts respectively for each Ice Lake, Zen 3, and Altra core. The cores are all run at 2.4 5GHz. Throughput is measured for a single query (i.e., for data, media, web services) or entire workload execution time (i.e., for analytics).

The figure indicates that due to its more efficient x86 core design, Zen 3 on average improves both metrics over Ice Lake. Altra, utilizing a narrower out-of-order ARM pipeline, improves performance density by up to 4x and performance/power by up to 2.65x. These numbers for Ampere are conservative because Altra uses stock ARM cores. AmpereOne with custom ARM cores for servers is slated to have higher silicon efficiencies.

IV. Emerging Workloads & Paradigms

At Google, Luiz helped architect the *scale-out* workload paradigm. Scale-out emerged from the need to cost-effectively deliver online services to a global base of users. By their very nature, these services were complex (e.g., web search) yet needed to meet tight latency constraints and unprecedented throughput objectives. Luiz and colleagues demonstrated that scalability can be achieved by decomposing complex applications into stand-alone logical components, each of which can be deployed and scaled independently to offer the desired performance and reliability objectives in software.

While the scale-out paradigm enabled greater degrees of scalability and modularity compared to traditional enterprise *monoliths*, typical scale-out architectures featured just two or three tiers, with each tier responsible for a significant degree of the overall functionality. With increasing complexity and commoditization of online services, there emerged a need for greater modularity and elasticity at the software level, giving rise to microservices.

Microservices ushered in an era of truly disaggregated applications, each deployed as a graph of tens or even hundreds of nodes communicating via remote procedure calls (RPC). The high degree of modularity endemic in microservices facilitated composability, whereby off-the-shelf pieces of software (potentially written in different programming languages) could be mixed and matched to rapidly engineer and deploy complex services. High modularity also enabled greater degrees of elasticity, empowering developers to scale different parts of their application separately as a function of the load placed on each microservice.

A performance analysis of microservices by Gan et al. [6] observed that the network stack accounted for a significant fraction of cycles, due to frequent RPC calls – the price of disaggregation. The trends at the microarchitectural largely echoed those for scale-out and enterprise workloads, with a high fraction of stall cycles in the core front-end and long-latency accesses to main memory for data, both of which contribute to low IPC [5]. Microservices also amplify the tail latency problem, because a single slow server in a long chain of RPC calls can compromise the end-to-end latency.

In recent years, *serverless* computing has emerged as a popular way to deploy complex services in the cloud. Similar to microservices, a serverless application comprises a graph of loosely connected functions that communicate over RPC. In serverless, however, functions are stateless, with any mutable state maintained in a separate service.

Serverless offers solutions to key drawbacks of microservices. First, traditional microservice instances are often housed within full-featured virtual machines (VMs) that can take considerable time to boot. This prolonged boot time can compromise latency targets, especially during sudden load spikes. Developers often provision additional VM instances as a precaution against load fluctuations, leading to overprovisioning and unnecessary costs as VMs are billed regardless of whether they are idle or active.

Serverless addresses these challenges by leveraging its stateless nature and lightweight *micro-VMs*, facilitating rapid on-demand scaling with instance boot times measured in seconds rather than minutes. Moreover, developers are billed solely for the actual runtime and memory usage of their functions, encouraging them to optimize instances' runtime and memory footprints. Consequently, serverless functions exhibit short execution times, often in milliseconds, and small memory footprints [10], enhancing server utilization through efficient bin packing. However, as serverless adoption increases, it will necessitate adjustments in OS and hardware to efficiently handle the growing number of tasks and remote data access, ensuring optimal performance and resource utilization while managing potential inefficiencies in the software stack.

AI workloads represent a substantial portion of datacenters' computational resources and energy consumption, with anticipated growth driven by the rise of generative AI technologies. To address these requirements, major cloud providers such as Google, Microsoft, Amazon, and Huawei have introduced AI accelerators, offering higher silicon efficiency compared to CPUs. AI workloads also align with trends favoring disintegration for the scalability of services and resource optimization, leading to their deployment as microservices and serverless instances. However, this trend also underscores the need for solutions capable of managing diverse requirements for computational resources, while also supporting frequent use of OS (e.g., memory management and scheduling) and network services (e.g., RPC).

V. Post-Moore Servers

There are many opportunities to optimize silicon efficiency in the post-Moore era. These opportunities can bridge the gap between the workloads and emerging silicon technologies to minimize both capital expenditure (i.e., amount of silicon used) and operational expenditure (i.e., energy used) in datacenters [3].

Today's servers inherit their hardware and operating system from the desktop PCs of the 1990s, where silicon is fragmented across the hardware and OS boundary lines. This architecture fundamentally suffers from the same mismatch between the nanosecond time-scale of CPU and memory with the millisecond time-scale of OS services handling I/O devices (mentioned in II). Because datacenters' primary function is processing, managing and storing distributed data, to access this data hosted in the server's memory requires CPU and OS intervention. Moreover, discrete I/O devices require their own memory (and CPU in the case of network interface cards) fragmenting silicon across the server and creating multiple copies of data. Finally, because the flow of data among services is not exposed to the OS within the server, the OS is oblivious to how CPU resources are scheduled and is unable to minimize OS scheduling overhead and maximize instruction and data affinity.

To reduce data's overall required silicon footprint, there are three post-Moore design pillars for servers: (1) *integration*, tightly connecting functionality with data in memory to reduce movement, (2) *specialization*, tailoring silicon resources for common functionality for data, and (3) *approximation*, optimize silicon resources based on the expected output quality for computation. In the following, we present examples of opportunities to pursue with these pillars to optimize silicon efficiency in post-Moore servers.

A. Compute

While silicon density continues to improve slowly with new fabrication technologies, these technologies are becoming increasingly expensive. Chiplets have emerged as a promising approach to reduce the cost of fabrication both through enhancing yield (because yield is a function of die area) and the use of older technology nodes for the less latency-critical functionality. Chiplets also present an opportunity to build a single server node with disaggregated specialized functionality and tighter integration to reduce data movement across services. Services with large instruction supply bottlenecks, large off-chip datasets, and little instruction- and memory-level parallelism can use a scale-out pod [9] chiplet architecture with more logic-to-memory silicon provisioning. To accommodate a larger number of cores per chiplet, chiplet design can be tailored with a minimal frequency to achieve the desired tail latency in a disaggregated service. Microservices and serverless workloads would benefit from a chiplet architecture with hardware support for core-to-core connectivity, RPC operations and a tighter integration with the network interface.

Moreover, the energy required to fetch a word of off-chip memory is four or more orders of magnitude larger than a fixed-point add operation. Popular analytic operators (e.g., in Apache Spark) filter, map, aggregate, join, or sort data in memory which is distributed over multiple servers. These operators are embarrassingly parallel, require only fixed-point arithmetic, can easily consume the available memory bandwidth, and do not exhibit much reuse in the cache hierarchy. Custom chiplets for analytic operators can be properly integrated both near memory to utilize the available memory bandwidth and near the network interface card to stream the results out in case of a request from a remote server.

Finally, numerical encoding is a fundamental enabler in reducing data's overall computational, communication and storage requirements. There is a difference of two orders of magnitude in energy and area footprint for 2-bit vs. 32-bit arithmetic operations. Much computation centered around data operates on data that is statistical in nature (e.g., originating from sensors or simulation). Similarly, computational outcome is often statistical in nature, gauged by a given quality metric (e.g., training, inference, statistical ranking, media). Identifying algorithmic techniques for approximation together with innovation in numerical encoding and microarchitecture can help maximize computational density, and minimize the required memory capacity and bandwidth. Modern GPUs are increasingly supporting both multiple scaled numerical formats and sparsity to allow algorithms to minimize the computational, memory capacity and bandwidth requirements of large language models.

B. Memory

Memory is roughly 50% of the cost of the server today and has been highly cost-sensitive since the inception of datacenters. In recent years, the cost per GB has plateaued due to the slowdown in Moore's Law, further pushing up the overall memory price per server. There are many opportunities to explore technologies that enable both a more efficient use of memory and its capacity, directly improving the return on investment in servers.

Memory pooling is one such paradigm that has great potential. Many studies in the past have examined rack-level memory pooling to mitigate a load imbalance in data serving software stacks due to a skew in partitioning or object popularity. Recent research suggests that, in the cloud environment, memory is wasted in rented VMs because customers don't fully use the memory capacity, and in rented physical servers where all CPUs are rented and memory can be stranded. Tighter integration of network interface with the CPU would enable the OS to implement memory pooling and optimize capacity management. CXL also offers a promising scalability path for memory pooling with transaction-oriented serial links in future generation PCIe protocols as traditional parallel interfaces in DDR are prohibitively pin-intensive.

Virtualization is a foundational technology built on top of virtual memory to provide isolation for both conventional cloud containers and emerging serverless/microservices paradigms. Confidential computing also fundamentally relies on virtualization for protection. Modern page-based virtual memory dates back to the 1960s and its current incarnations not only incur significant silicon footprint for TLBs (now in thousands of entries per core) but also operate on a microsecond time-scale for page-based memory management operations (e.g., changing permissions).

Intermediate address spaces are a promising approach to enabling nanosecond-scale memory management operations [7] while eliminating the silicon footprint of page-based virtual memory. Emerging workloads such as microservices, serverless functions and virtualized network functions can forego virtualization and specialize hardware/software for nanosecond-scale compartments using an intermediate address space. The ability to grant and revoke access to memory on the nanosecond time-scale means that data can be supplied from across multiple domains from the same physical copy, thereby eliminating data copy operations and providing a tighter integration between the application and the system.

C. Datacenter Tax

A common set of operations in datacenters accounts for 30% of all CPU cycles. These operations, referred to as *datacenter tax*, are centered around the management and communication of data. These include CPU cycles in the datacenter spent moving and operating data among the CPU, memory, and the network. Both DDR and CXL (with PCIe) bandwidth are slated to grow by more than 20% per year by 2030. Network bandwidth is also slated to grow by 20% in the next decade thanks to emerging fabrics. With these bandwidth trends, the sequential software running on the CPU to perform the datacenter tax will be a growing bottleneck. Mitigating the datacenter tax requires proper integration of specialized logic among the CPU, memory, and the network to accelerate these operations and minimize data movement.

Many common datacenter tax operations are related to memory management. These include memory allocation, copying, compression/decompression, encryption/decryption, and scatter/gather operations. Because RPC is the most common mechanism of communication, the datacenter tax also includes data serialization/deserialization to allow microservices to translate their object formats in memory as they communicate. Recent products such as Intel Sapphire Rapids started incorporating examples of such accelerators.

There are promising avenues to co-design custom processors for RPC [12]. These processors, when integrated with the network interface and the CPU in the same coherence domain,

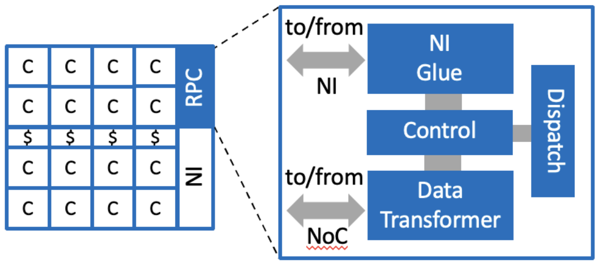

Fig. 4. The anatomy of the Cerebros RPC processor.

can minimize data movement. Figure 4 depicts the anatomy of the Cerebros RPC processor tightly integrated with the network interface logic and CPU. The processor executes the RPC layer (e.g., Apache Thrift or Google Protobuf) and acts as an intermediary stage between the network interface and the microservice running on the CPU. The processor not only processes RPC requests at line rate (e.g., 100 Gbps), it also optimizes service throughput by monitoring and balancing load on cores, and using affinity scheduling to improve locality.

Beyond memory allocation and management, common OS operations such as thread scheduling and interrupt delivery also consume a significant fraction of CPU cycles. Modern servers consolidate thousands of threads, which the OS has to manage within a coherence domain. Most importantly, the OS cannot maximize affinity and minimize data movement and interrupt delivery overhead because it has little information about the application's communication patterns. The combined effect of increasing core counts and emerging paradigms (i.e., microservices and serverless computing) exacerbate the overhead of thread management, scheduling and interrupt delivery. Innovation in OS together with microarchitecture will be instrumental in mitigating the OS-centric datacenter tax.

VI. Conclusions

Server architecture has undergone major transformations from the enterprise servers of the 1990s to the cloud servers of today. These transformations have been the thanks to exponential improvements in CPU performance and memory density with Moore's Law, the demand for scalability from enterprise IT to public cloud, and the focus on return on investment in datacenters. Luiz played an indispensable role in creating and realizing these transformations. Continuing Luiz's legacy in the post-Moore era requires a clean-slate approach to the design of the computing stack, from algorithms all the way down to hardware, to maximize silicon efficiency and address the needs of emerging workloads with proper design metrics.

References

[1] L. Barroso, M. Marty, D. Patterson, and P. Ranganathan, "Attack of the killer microseconds," *Commun. ACM*, vol. 60, no. 4, 2017.
[2] L. A. Barroso, K. Gharachorloo, and E. Bugnion, "Memory system characterization of commercial workloads," in *Proceedings of the 25th International Symposium on Computer Architecture*, 1998.
[3] L. A. Barroso, U. Hölzle, and P. Ranganathan, *The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2018.
[4] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, 2013.
[5] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Quantifying the mismatch between emerging scale-out applications and modern processors," *ACM Transactions on Computer Systems*, vol. 30, no. 4, 2012.
[6] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson *et al.*, "An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.
[7] S. Gupta, A. Bhattacharyya, Y. Oh, A. Bhattacharjee, B. Falsafi, and M. Payer, "Rebooting virtual memory with midgard," in *Proceedings of the 48th International Symposium on Computer Architecture*, 2021.
[8] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, no. 4, 2011.
[9] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-Out Processors," in *Proceedings of the 39th International Symposium on Computer Architecture*, 2012.
[10] M. Shahrad, R. Fonseca, I. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *USENIX Technical Conference*, 2020.
[11] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe, "Simflex: Statistical sampling of computer system simulation," *IEEE Micro*, vol. 26, no. 4, 2006.
[12] A. P. Zarandi, M. Sutherland, A. Daglis, and B. Falsafi, "Cerebros: Evading the RPC tax in datacenters," in *Proceedings of the 54th IEEE/ACM International Symposium on Microarchitecture*, 2021.