Building High-performance and Versatile Dataplane for Modern Datacenter Network

Research Statement

Yifan Yuan Intel Labs yifan.yuan@intel.com https://yifanyuan3.github.io

In the past decade, the datacenter network has evolved at a fast pace. Currently, 100 Gbps Ethernet is widely adopted by datacenters, and 400 Gbps Ethernet is not far on the horizon. On the one hand, the slow-down of Moore's Law generates an increasingly large gap between the network processing capability on the conventional hardware system and the modern network performance. On the other hand, emerging networking devices tend to be rich with accelerators and programmable logics, offloading software stacks and applications. This calls for significant hardware and software innovations to build a high-performance datacenter. My research aims to achieve this goal from two aspects: **(A)** enhancing the existing system components (e.g., CPU, network stack) to handle the fast network efficiently; **(B)** leveraging the network itself to accelerate a broader range of applications.

Current Research

My current research has covered these two aspects from multiple dimensions and led to eight peer-reviewed publications in top-tier conferences related to computer architecture and systems [1–8], as well as four filed US patents [9–12]. Some of the research outcomes have been transferred/ are being transferred to industry products [3, 4, 6]. I categorize my research into three thrusts. For aspect-**A**, we **(1)** consider the ways of better utilizing and managing existing hardware features (i.e., performance optimization and tuning) and **(2)** try to add new components (i.e., hardware accelerator) to further boost the performance of datacenter applications and infrastructures. For aspect-**B**, taking distributed machine learning training as a representative workload, we **(3)** leverage the networking devices, including NICs (for in-network gradient compression) and switches (for in-network gradient aggregation), to facilitate the inter-machine communication, which is the most expensive part in distributed training.

1. Adaptive Interaction across CPU, Memory Hierarchy, and I/O (Networking) Devices

IAT [3]. In modern server CPUs, last-level cache (LLC) is a critical hardware resource that exerts significant influence on the performance of the workloads, and how to manage LLC is a key to the performance isolation and QoS in the datacenter with multi-tenancy. We argue that in addition to CPU cores, high-speed I/O is also important for LLC management. This is because of an Intel architectural innovation – Data Direct I/O (DDIO) – that directly injects the inbound I/O traffic to (part of) the LLC instead of the main memory. We summarize two problems caused by DDIO and show that (1) the default DDIO configuration may not always achieve optimal performance, (2) DDIO can decrease the performance of non-I/O workloads that share LLC with it by as high as 32%. We then present IAT, an I/O-aware LLC management mechanism. IAT monitors and analyzes the performance of the core/LLC/DDIO using CPU's hardware performance counters and adaptively adjusts the number of LLC ways for DDIO or the tenants that demand more LLC capacity. In addition, IAT dynamically chooses the tenants that share its LLC resource with DDIO to minimize the performance interference by both the tenants and the I/O. Our experiments with multiple microbenchmarks and real-world applications demonstrate that with minimal overhead, IAT can effectively and stably reduce the performance degradation caused by DDIO.

Impact: IAT is the **first** LLC management mechanism that treats the I/O as a first-class citizen. The corresponding software APIs will be upstreamed to Intel official RDT library (pqos) [13] in its next release.

Besides, although effective on DRAM-based memory, DDIO does not always improve the performance of NVM-based memory (e.g., Intel Optane DIMM). This is mainly because of the data persistence requirement and write-amplification phenomenon. We propose a device-oriented approach to control the DDIO behavior in a system with both DRAM and NVM with no hardware modification [14]. To overcome the inflexibility of the commodity hardware, based on our performance and behavior characterization, we redesign the I/O subsystem model in gem5, a popular cycle-accurate full-system simulator, to facilitate DDIO research in the architecture and systems communities [5].

2. Generic and Efficient Accelerators for Dataplane Operations

QEI [4]. Data query operations of different data structures are ubiquitous and critical in today's datacenter's network dataplane (both infrastructures and applications). However, query operations are not always performance-optimal to be executed on general-purpose CPU cores. These operations exhibit insufficient memory-level parallelism and frontend bottlenecks due to unstructured control flow. Furthermore, the data access patterns are not cache- or prefetch-friendly. Based on our performance analysis on a commodity server, query operations can consume a large percentage of the CPU cycles in various modern cloud workloads. Existing accelerator solutions for query operations do not strike a balance between their generality, scalability, latency, and hardware complexity. We propose QEI, a generic, integrated, and efficient acceleration solution for various data structure queries. We first abstract the query operations to a few regular steps and map them to a simple and hardware-friendly configurable finite automaton model. Based on this model, we develop the QEI architecture that allows multiple query operations to execute in parallel to maximize throughput. We also propose a novel way to integrate the accelerator into the CPU that balances performance, latency, and hardware cost. QEI keeps the main control logic near the L2 cache to leverage existing hardware resources in the core while distributing the data-intensive comparison logic to each last-level cache slice for higher parallelism. Our results with five representative datacenter workloads show that QEI can achieve 6.5× ∼ 11.2× performance improvement in various scenarios with low overhead.

Impact: With the corresponding patent on file, QEI's design methodology is being transferred to Intel's product team, and is on the path of being implemented in future Xeon CPUs.

In addition, we also design HALO [6], a customized case of QEI that is specifically optimized for the software virtual switch in network function virtualization (NFV) environment. Experiments with a variety of real-world workloads of network services demonstrate that HALO improves the throughput of basic flow-rule lookup operations by 3.3×, and scales the representative flow classification algorithm – tuple space search by up to 23.4× with negligible negative impact on the performance of collocated network services, compared with state-of-the-art software-based solutions.

3. In-network Computing for distributed Applications

FPISA [2]. The advent of switches with programmable dataplanes has enabled the rapid development of new network functionality, as well as providing a platform for acceleration of a broad range of application-level functionality. However, existing switch hardware was not designed with application acceleration in mind, and thus applications requiring operations or datatypes not used in traditional network protocols must resort to expensive workarounds. Applications involving floating point data, including distributed training for machine learning and distributed query processing, are key examples. We propose FPISA, a floating point representation designed to work efficiently in programmable switches. We first implement FPISA on an Intel Tofino switch, but find that it has limitations that impact throughput and accuracy. We then propose hardware changes to address these limitations based on an open-source programmable switch architecture, and synthesize them in a 15-nm standard-cell library to demonstrate their feasibility. Finally, we use FPISA to implement accelerators for training for machine learning and for query processing, and evaluate their performance on a switch implementing our changes using emulation. We find that FPISA allows distributed training to use 25∼75% fewer CPU cores and provide up to 85.9% better throughput in a CPU-constrained environment than the state-of-the-art work. For distributed query processing with floating point data, FPISA enables up to 2.7× better throughput than Spark.

Impact: FPISA opens a new direction for modern programmable switches that deals with new and important datatypes flexibly and efficiently. Any networking/distributed applications involving floating-point operations can benefit from FPISA, with little/no hardware modification.

FPISA is based on our previous exploration that moves the expensive gradient aggregation part of distributed ML training into the network [7]. Such an in-network aggregation approach can significantly reduce the network and compute bottlenecks of the parameter server in conventional distributed training. We also explore ring-allreduce topology for gradient aggregation, and propose a lightweight and hardware-friendly lossy-compression algorithm for floating-point gradients [8]. Exploiting the gradient's unique value characteristics, this compression not only enables significantly reducing the gradient communication with practically no loss of accuracy, but also comes with low complexity for direct implementation as a hardware block in the NIC. Our experiments demonstrate that our proposal reduces the communication time by 70.9∼80.7% and offers 2.2∼3.1× speedup over the conventional training system, while achieving the same level of accuracy.

Future Plan

Near-term Direction: Embracing the Emerging System Interconnects in the Datacenter

Recently, emerging system interconnects (e.g., CXL, Gen-Z, CAPI, CCIX) have created numerous opportunities for architecture and systems innovations in both industry and academia. Among them, CXL is the most popular and influential initiative supported by major hardware vendors, offering devices with break-through performance characteristics and features. I propose a three-phase research agenda to make the most out of CXL devices.

First, I plan to conduct a comprehensive characteristic study on the CXL-enabled commodity platform, with an emphasis on the difference between regular PCIe and CXL devices. This helps us and the community better understand the unique advantages and use cases of CXL.

Based on these characteristics study results, I plan to build hardware and system software for better CXL device support in the second phase of this research. On the hardware side, I plan to investigate and optimize the host-device interaction over CXL, including but not limited to memory/cache hierarchy (e.g., the effect and tuning of DDIO), notification based on the new cache coherence feature (CXL.cache), efficient virtual-physical address translation support (CXL.mem); on the software side, I plan to leverage such hardware features and optimizations to build more efficient device drivers and OS primitives. For example, in the disaggregated cloud, a new heterogeneous-ISA OS can be built for a system with x86 CPU as the host and ARM/RISC-V CPU on a CXL.cache device, providing a paradigm where the beefy x86 cores are a type of resource and the wimpy ARM/RISC-V cores can be responsible for resource scheduling, workload dispatching, etc.

In the last research phase, based on the previous research results, I plan to find specific applications and OS routines that are suitable to be offloaded/facilitated by co-designing host and CXL devices. Such co-design targets a wide range of cloud environments, from machine learning, micro-service, to resource disaggregation. Significant performance and energy-efficiency improvement can be anticipated.

My previous research experience based on another cache-coherent interconnect [14] has illustrated its promising benefits of solving the "datacenter tax" problems and "killer microsecond" problem. I will continue this direction to build high-performance and efficient datacenter hardware and system software.

Long-term Direction: Toward Energy Efficiency in the heterogeneous Datacenter

Having discussed all the compute resources in the modern datacenter with high heterogeneity, including but not limited to conventional CPUs, on-chip and off-chip accelerators, networking devices, I plan to investigate another crucial aspect of the datacenter – energy efficiency – from a holistic perspective. On a global scale, datacenters can consume 416 terawatts of power, or roughly 3% of all electricity generated all over the world, and any power reduction in the datacenter can result in significant benefits to our planet. Specifically, an application (or infrastructure) can run on different compute resources with different performance. With a given performance target (e.g., service-level agreement or SLA), where is the best place to allocate this application with low power consumption? In other words, how can we optimally leverage the heterogeneous resources in the datacenter to achieve the performance target? This problem requires multi-dimensional research effort, ranging from datacenter resource usage analysis, to application/infrastructure performance characterization and modeling, scheduling/allocation algorithm/policy design, architectural enhancement and tuning, and prototyping and deployment.

Based on my previous and current experience in system modeling [5], application analysis and acceleration [4, 6], and resource management [3], I am well prepared to conduct such full-stack and cross-domain research with researchers/engineers from different teams.

Services and Outreach

I plan to extend my effort in community services and outreach in the future. I have served in the (external) program committees of EuroSys'22 and HPCA'23, mentored a diversity of graduate and undergrad students to facilitate their early-stage research. In addition to continuously serving in the program committees of the related conferences/journals, I will be committed to the open-source and artifact evaluation trend in architecture and systems research – this will not only improve the impact of the works, but also facilitate the subsequent research activities in the community. Also, I will actively get in touch and collaborate with Intel's product teams. On the one hand, this helps transfer the research prototypes into real-world deployments. On the other hand, I would like to identify and solve the technical challenges in the real datacenter environment.

References

[1] M. Alian, S. Agarwal, J. Shin, N. Patel, Y. Yuan, D. Kim, R. Wang, and N. S. Kim. IDIO: Network-driven, inbound network data orchestration on server processors. In *Proceedings of the 55th the ACM/IEEE International Symposium on Microarchitecture (MICRO'22)*, 2022.

[2] Y. Yuan, O. Alama, J. Fei, J. Nelson, D. R. K. Ports, A. Sapio, M. Canini, and N. S. Kim. Unlocking the power of inline floating-point operations on programmable switches. In *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22)*, 2022.

[3] Y. Yuan, M. Alian, Y. Wang, R. Wang, I. Kurakin, C. Tai, and N. S. Kim. Don't forget the I/O when allocating your LLC. In *Proceedings of the 48th IEEE/ACM International Symposium on Computer Architecture (ISCA'21)*, 2021.

[4] Y. Yuan, Y. Wang, R. Wang, R. B. R. Chowhury, C. Tai, and N. S. Kim. QEI: Query acceleration can be generic and efficient in the cloud. In *Proceedings of the 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA'21)*, 2021.

[5] M. Alian, Y. Yuan, J. Zhang, R. Wang, M. Jung, and N. S. Kim. Data direct I/O characterization for future I/O system exploration. In *Proceedings of the 2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'20)*, 2020.

[6] Y. Yuan, Y. Wang, R. Wang, and J. Huang. HALO: Accelerating flow classification for scalable packet processing in NFV. In *Proceedings of the 46th IEEE/ACM International Symposium on Computer Architecture (ISCA'19)*, 2019.

[7] Y. Li, I.-J. Liu, Y. Yuan, D. Chen, A. Schwing, and J. Huang. Accelerating distributed reinforcement learning with in-switch computing. In *Proceedings of the 46th International Symposium on Computer Architecture (ISCA'19)*, 2019.

[8] Y. Li, J. Park, M. Alian, Y. Yuan, Z. Qu, P. Pan, R. Wang, A. G. Schwing, H. Esmaeilzadeh, and N. S. Kim. A network-centric hardware/algorithm co-design to accelerate distributed training of deep neural networks. In *Proceedings of the 51st International Symposium on Microarchitecture (MICRO'18)*, 2018.

[9] R. Wang, T.-Y. C. Tai, Y. Wang, Y. Yuan, S. Paul, M. M. Khellah, S. Gobriel, C. Augustine, M. Ganguli, J.-S. Tsai, E. Verplanke, P. Autee, A. Layek, S. Narayana, B. Ganesh, J. B. Timbadiya, S. K. Muthukumar, R. Iyer, N. Jain, N. D. McDonnell, M. A. Goldschmidt, R. M. Sankaran, and N. Ranganathan. Hardware assisted lookup operations, 2020. US Patent App. 63/130,663.

[10] R. Wang, Y. Yuan, Y. Wang, T.-Y. C. Tai, and T. Hurson. Data consistency and durability over distributed persistent memory systems, 2020. US Patent App. 62/986,094.

[11] Y. Wang, R. Wang, T.-Y. C. Tai, Y. Yuan, P. Pathak, S. Vedantham, and C. Macnamara. Workload scheduler for memory allocation, 2020. US Patent App. 16/799,745.

[12] R. Wang, A. J. Herdrich, T.-Y. C. Tai, Y. Wang, R. Kondapalli, A. Bachmutsky, and Y. Yuan. Offload of data lookup operations, 2018. US Patent App. 16/207,065.

[13] Intel Corporation. User space software for Intel resource director technology.
https://github.com/intel/intel-cmt-cat.

[14] Y. Yuan, J. Huang, Y. Sun, T. Wang, J. Nelson, D. R. K. Ports, Y. Wang, R. Wang, C. Tai, and N. S. Kim. ORCA: A network and architecture co-design for offloading us-scale datacenter applications. In *Under review*, 2021.