Research Statement

Yifan Sun

My research focuses on enabling and scaling the performance of future computing systems that consist of a large number of highly-diverse computing devices. Today, computing is an essential resource that drives the development of science and technology. The recent development in both high-performance and low-power computing significantly improves critical technologies such as remote sensing, medical imaging, computational materials science, and artificial intelligence. Considering that modern applications solve a wide range of problems, a single type of devices, such as CPUs, GPUs, or accelerators, are not likely to run all applications efficiently. Future computing systems need to employ diverse computing devices, supported by efficient communication mechanisms. My previous work focused on CPU-GPU and multi-GPU collaborative computing. In the future, I plan to explore Fabric-Centric Computing, a holistic, cross-stack, cross-layer approach that considers power, cost, privacy, and security. My solutions will support the collaboration of various types of computing devices in edge nodes, data-center nodes, and the network.

1. Contributions

My previous research has focused on enabling GPUs to execute collaboratively with CPUs and other GPUs. GPUs can deliver impressive computing power thanks to their massively parallel architectures. However, single-GPU systems cannot meet the demands of modern workloads. Specifically, low-power devices present a mixture of highly-parallelized, highly-serialized, and IO-heavy applications, requiring close collaboration between CPUs and GPUs. High-performance computing applications increasingly demand high performance, requiring multiple GPUs to compute the solution for a single application. Existing software and hardware solutions do not typically support efficient CPU/GPU communication. To improve collaboration across multiple devices, I pursue full-stack research that covers workload development, performance modeling, and software/hardware designs that improve system efficiency in both CPU-GPU and Multi-GPU Collaborative Computing.

1.1. CPU-GPU Collaborative Computing

Current GPU workloads rarely allow CPUs and GPUs to execute simultaneously, resulting in the underutilization of resources. To explore the design space of CPU-GPU simultaneous execution, I have developed a taxonomy of 11 common CPU-GPU collaborative computing patterns to guide future software and hardware designs. I have led the development of Hetero-Mark, a benchmark suite that explores CPU-GPU collaborative computing patterns [ISPASS15, IISWC16, ISPASS18]. To facilitate CPU-GPU collaborative computing performance analysis, I have designed and developed Multi2Sim-HSA, an HSA instruction-level emulator based on the Multi2Sim simulator [BookChapter]. I have also contributed to system designs that improve CPU-GPU collaborative execution, including Unified Memory Hierarchy (UMH) and Airavat. UMH utilizes GPU DRAMs as caches of system memory and unifies the CPU and GPU memory hierarchy [TACO16]. Airavat coordinates the frequency of CPUs, GPUs, and DRAMs, reducing conflicts between the CPU cores and GPU cores and improving energy efficiency [DATE18].

1.2. Multi-GPU Collaborative Computing

Utilizing multiple GPUs has become a standard solution to deliver higher computing throughput. However, research on multi-GPU system performance is still in its early stages due to a lack of multi-GPU benchmark suites and multi-GPU simulators. To address this research gap, in MGPUMark, I have characterized various multi-GPU communication patterns and explored multi-GPU collaborative execution by developing a set of multi-GPU workloads. To advance multi-GPU research, I have initiated and led the development of a multi-GPU simulator--MGPUSim--featuring high-flexibility, high-performance, parallel, multi-GPU simulation [ISCA19]. To improve multi-GPU system performance and reduce inter-GPU communication, I have proposed several solutions to reduce inter-GPU traffic, including the Locality API, Progressive Page-Splitting Migration (PASI), and Adaptive Compression. The Locality API mechanism allows the explicit specification of data and thread placement to eliminate redundant inter-GPU traffic. PASI enables hardware to leverage runtime information to adjust data placement according to thread placement adaptively. Adaptive Compression allows GPUs to compress inter-GPU traffic. By learning the inter-GPU traffic data pattern, Adaptive Compression can select the best compression algorithm to balance the compression latency and the resulting compression rate [IPDPS19].

2. Future Directions

Emerging technologies are reforming the landscape of modern computing systems. First, emerging specialized accelerators such as video encoders/decoders, encryption/decryption engines, and Convolutional Neural Network accelerators can deliver much higher computing performance per watt than traditional devices. Second, emerging memory technologies such as High-Bandwidth Memory (HBM), Non-Volatile Memory (NVMe), and Phase-Change Memory (PCM) can potentially improve system performance and simplify software development. Finally, with the development of network technologies such as multi-chip-module communications, silicon-photonics, and 5G, computing devices are closely connected like never before.

My vision for future computing systems comprises highly diverse computing devices and memory modules that are interconnected by various types of networks. Performance improvements for computing systems have become a complex networking problem that involves network-on-chip, chip-to-chip, and node-to-node interconnects. Networking plays an even more critical role in holistic computing system designs that consider edge devices and data-center nodes. Future computing system design should thoughtfully include the network system design and should include interconnections as a critical design consideration.

To enable future hyper-connected computing systems, I plan to explore Fabric-Centric Computing, where the computing system design should prioritize data transfer efficiency over the communication fabric. Fabric-Centric Computing highlights cross-layer and cross-stack designs (i.e., the computing stack represented by compilers, ISAs, and microarchitectures and the network stack represented by data-link layer protocols and routing algorithms) that consider low-level microarchitecture design with low-level network protocol implementation. This change will require new software, hardware, and network co-designed to scale system performance. Fabric-Centric Computing also highlights the capability of the fabric to process data and ensure security. In particular, I plan to explore Fabric-Centric Computing while pursuing the following issues.

2.1. Designing Cross-stack Communication and Memory Management Protocols

The main goal of existing communication protocols is to deliver packets to their destinations as fast as possible. However, as communication protocols know little about the running application, they cannot schedule data transfers intelligently to improve overall system performance. Meanwhile, existing memory management protocols (e.g., cache coherency protocols) assume all memory modules at the same level in the memory hierarchy are equivalent, ignoring complex network topologies. Separating communication protocol design and memory management protocol design will result in performance limitations due to network congestion or cache thrashing. Co-designed communication and memory management protocols offer solutions that bridge the gap between the communication protocol stack and the computing stack. As a short term goal, I am developing a set of Collective Communication-Aware Cache Coherency protocols that allow multiple GPUs to overlap execution with communication. In the future, I plan to design communication-aware memory management protocols and computing-aware communication protocols that further improve the performance of future computing systems.

2.2. Utilizing Data-Path-Level Parallelism in Computing System Design

Future computing systems will incorporate a large number of compute cores and multiple intermediate devices that feed data to the compute cores. As the communication devices can transfer data in parallel, data processing on the communication path can also run in parallel. To utilize Data-Path-Level parallelism, I am planning to design special-purpose caches and switches that can process data in parallel while moving the data to the destination. My vision also includes architecting general-purpose pipe-based computers that minimize data movement through communication channels and memory hierarchy.

2.3. Crafting Highly-Usable Simulation Infrastructure

Developing highly-usable simulators that promote the development of the computer system research community has been, and will always be, one of my passions. Future research on Fabric-Centric Computing demands robust simulation infrastructures that combine both the computer network domain and the computer architecture domain. Moreover, developing simulators serves as an integral part of my education plan, as developing simulator components is an excellent approach for students to build a solid understanding of how computers work and communicate. As I lead large-scale open-source software developments, students will gain valuable experience from working in the open-source community.